First Hit Fwd Refs

☐ Generate Collection Print

L14: Entry 12 of 13 File: USPT Jan 18, 2000

US-PAT-NO: 6016486
DOCUMENT-IDENTIFIER: US 6016486 A

TITLE: System method and article of manufacture for a goal based system utilizing an activity table

DATE-ISSUED: January 18, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Nichols; Mark Stewart	Downers Grove	IL		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
AC Properties B.V.				NL	03

APPL-NO: 09/ 218968 [PALM]
DATE FILED: December 22, 1998

INT-CL: [06] G06 F 17/00

US-CL-ISSUED: 706/47; 705/40, 434/118
US-CL-CURRENT: 706/47; 434/118, 705/40

FIELD-OF-SEARCH: 705/40, 406/11, 406/47, 434/118

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4622013	November 1986	Cherchio	434/118
☐	4977529	December 1990	Gregg et al.	
☐	5002491	March 1991	Abrahamson et al.	
☐	5189402	February 1993	Naimark et al.	
☐	5267865	December 1993	Lee et al.	
☐	5310349	May 1994	Daniels et al.	
☐	5372507	December 1994	Goleh	
☐	5441415	August 1995	Lee et al.	

☐	5533903	July 1996	Kennedy	
☐	5537141	July 1996	Harper et al.	
☐	5539869	July 1996	Spoto et al.	
☐	5566291	October 1996	Boulton et al.	
☐	5576844	November 1996	Anderson et al.	
☐	5577186	November 1996	Mann, II et al.	
☐	5597312	January 1997	Bloom et al.	
☐	5616033	April 1997	Kerwin	
☐	5644686	July 1997	Hekmatpour	
☐	5644727	July 1997	Atkins	705/40
☐	5673369	September 1997	Kim	
☐	5690496	November 1997	Kennedy	
☐	5696885	December 1997	Hekmatpour	
☐	5720007	February 1998	Hekmatpour	
☐	5727950	March 1998	Cook et al.	
☐	5772446	June 1998	Rosen	
☐	5788508	August 1998	Lee et al.	
☐	5791907	August 1998	Ramshaw et al.	
☐	5799292	August 1998	Hekmatpour	
☐	5806056	September 1998	Hekmatpour	
☐	5810747	September 1998	Brudney et al.	
☐	5822745	October 1998	Hekmatpour	
☐	5823781	October 1998	Hitchcock et al.	
☐	5823788	October 1998	Lemelson et al.	
☐	5835683	November 1998	Corella et al.	
☐	5868575	February 1999	Kuczewski	
☐	5870768	February 1999	Hekmatpour	
☐	5875437	February 1999	Atkins	705/40
☐	5889845	March 1999	Staples et al.	

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 98/03953	January 1998	WO	

OTHER PUBLICATIONS

KBLPS Overview [online]. Carnegie Group, Inc., Jan. 1997 [retrieved on Aug. 12, 1999]. Retrieved from the Internet: <URL:www.cgi.com/CGIWEB/KBLPS/overindex4.html>.
MUSE U.S. Patents [online]. Occam Research Corporation, Jan. 1995 [retrieved on

Aug. 9, 1999]. Retrieved from the Internet: <URL:www.muser.com/html/patents.html>.
Vanguard Software Corporation, Decision Pro3.0 [online]. Vanguard Software Corporation, Jan. 1999 [retrieved on Aug. 9, 1999]. Retrieved from the Internet:<URL:www.vanguardsw.com/>.
FinPlan System [online]. Russian Research Institute of Artificial Intelligence, Jan. 1999 [retrieved on Aug. 9, 1999]. Retrieved from the Internet:<URL:www.rriai.org.ru/FinPlan/>.
Citation page for article published by ACM, the First Society in Computing. [online]. ACM Transactions on Information Systems, Jan. 1988 [retrieved on Aug. 9, 1999]. Retrieved from the Internet:<URL:www.acm.org/pubs/toc/Abstracts/tois/59298.html>.
Applications of BrainMaker Neural Network to stocks, business, medical, manufacturing [online]. California Scientific Software, Jan. 1999 [retrieved on Aug. 9, 1999]. Retrieved from the Internet:<URL:www.calsci.com/Applications.html>.
Rule-based Programming with OPS5[online]. Morgan Kaufmann Publishers, Jan. 1999 [retrieved on Aug. 9, 1999]. Retrieved from the Internet:<URL:www.mkp.com/books.sub.-- catalog/0-934613-51-6.asp>.
Brainmaker [online]. Echoscan Engineering Educational Equipments, Jan. 1999 [retrieved on Aug. 9, 1999]. Retrieved from the Internet: <URL:www.npiec.on.ca/.about.echoscan/28-04.htm>.
Automate Your Business Plan--financial statement ratio analysis screen capture [online]. Automate Your Business Plan, Jan. 1999 [retrieved on Aug. 9, 1999]. Retrieved from the Internet: <URL:www.business-plan.com/screen2.html>.
News for ESAP [online]. ESAP, Jan. 1999 [retrieved on Aug. 9, 1999]. Retrieved from the Internet: <URL:www.hops.wharton.upenn.edu/.about.esap/news.html>.
Useful Software [online]. University Of Maryland, Baltimore County, Jan. 1999 [retrieved on Aug. 9, 1999]. Retrieved from the Internet:<URL:http://research.umbc.edu/.about.malaga/ifsm625/software.html >.
Mark L. Lengnick-Hall, Virtual learning: A Revolutionary Approach to Building a Highly Skilled Workforce, Autumn 1998, Personnel Psychology vol. 51, No. 3, pp. 767-771.
J Bernard Keys, Robert M. Fulmer, Stephen A Stumpf, Microworlds and simuworlds: Practice fields for the learning organization, Spring 1996, Organizational Dynamics vol. 24, No. 4, pp. 36-49.
George Cole, Learning with Computers, May 1994, Accountancy vol.113, No. 1209, pp. 60-64.
Tony Burns, Multimedia training "Get lemonade, not a lemon!", Jun. 1997, Journal for Quality & Participation vol. 20, No. 3, pp. 22-26.
Lin Grensing-Pophal, Flexible learning, Feb. 1998, Credit Union Management vol. 21, No. 2, pp. 32-33+.
Charles W. Calmbacher, Interactive multimedia instructs the individual, Oct. 1994, Occupational Health & Safety vol.63, No. 10, pp. 144-145.
Jean-Francois Manzoni, Albert A. Angehrn, Understanding organizational dynamics of IT-enabled change: A multimedia simulation approach, Winter 1997/1998, Journal of Management Information Systems: JMIS, vol. 14, No. 3, pp. 109-140.
Marianne Kolbasuk McGee, Train with less pain, Oct. 13, 1997, Informationweek No. 652, pp. 150-154.
Engines for Education: URL: http://www.ils.nwu.edu/.about.e.sub.-- for.sub.-- e/nodes/I-M-Intro-Zoomer-pg.html; Viewed Feb. 15, 1999; Roger Schank; Web site claims 1994 copyright.
Socialized Collaborative Learning in Multimedia Virtual Worlds, Learning Environments, National University of Singapore; URL: http://www.iscs.nus.edu.sg/labs/learning/lels/VRML.html; Viewed Feb. 16, 1999.
C.D. Whittington, L. M. Campbell; Task-Oriented Learning on the Web; Innovations in Education and Training International; vol. 36, No. 1, Feb. 1999.
J. Gonzalez, J. V. Lopez, F. A. Bustio, P. Corcuera, E. Mora; Development of an Integrated Simulator and Real Time Plant Information System; Advances in the Operational Safety of Nuclear Power Plants, Proceedings of an International Symposium 1996.
Anup Kumar, Raj Pakala, R. K. Ragada, J. P. Wong; The Virtual Learning Environment

System; 28.sup.th Annual Frontiers in Education Conference; Conference Proceedings, vol. 2, Nov. 4-7, 1998.
Ramnath Chellappa, Anitesh Barua, Andrew B. Whinston; An Electronic Infrastructure for a Virtual University; Communications of the ACM, vol. 40, No. 9, Sep. 1997.
Wendy Doube; A Browser-based System to Support & Deliver DE; 28.sup.th Annual Frontiers in Education Conference; Conference Proceedings, vol. 1, Nov. 4-7, 1998.
David A. Foster; FRA: Teaching Financial Accounting with a Goal-Based Scenario; Intelligent Systems in Accounting, Finance and Management, vol. 4, 1995.
David A. Foster; Real-World Analysis Skills Goal-Based Scenario; pp. 68-74.
Allan Collins; Goal-Based Scenarios and the Problem of Situated Learning: A Commentary on Andersen Consulting's Design of Goal-Based Scenarios; Educational Technology, Nov.-Dec. 1994.
Jack. M. Wilson, David N. Mosher; Interactive Multimedia Distance Learning (IMDL): The Prototype of the Virtual Classroom; 1994.
Interview Conducted by William Graham; Goal-Based Scenarios and Business Training: A Conversation with Roger C. Schank; Educational Technology, Nov.-Dec., 1994.
Joel Montgomery, Robert Campbell, Christine Moffett; Conducting and Supporting a Goal-Based Scenario Learning Environment; Educational Technology, Nov.-Dec. 1994.
Alan Nowakowski; A Special Section--Goal-Based Scenarios: A New Approach to Professional Education: Reengineering Education at Andersen Consulting; Educational Technology, Nov.-Dec. 1994.
Al Seagren, Britt Watwood; The Virtual Classroom: Great Expectations. Delivering Graduate Education by Computer: A Success Story; 1996.
J.R. Anderson and B.J. Reiser, "The Lisp Tutor," Byte, pp. 159-175 Apr. 1985.
N.D. Livergood, "From Computer-Assisted Instruction to Intelligent Tutoring Systems," J.O Artificial Intelligence in Education, vol. 2(3), pp. 39-50 Dec. 1997.

W. Regian and G. Pitts, "A Fuzzy Logic-Based Intelligent Tutoring System," Information Processing 92, vol. II, pp. 66-72 Dec. 1992.
A.J. Gonzalez and L.R. Ingraham, "Automated Exercise Progression in Simulation-Based Training," IEEE Trans. On Systems, Man, and Cybernetics, vol.24(6), pp. 836-874 Jun. 1994.
R.H. Kemp and S.P. Smith, "Using Planning Techniques to Provide Feedback in Interactive Learning Environments," Proc. Sixth Int'l. Conf. On Tools with Artificial Intelligence, pp. 700-703 Nov. 1994.
J. Siemer and M.C. Angelides, "Embedding an Intelligent Tutoring System in a Business Graming-Simulation Environment," Proc. Of the 1994 Winter simulation Conference, pp. 1399-1406 Dec. 1994.
V.J. Shute, "Smart Evaluation: Cognitive Diagnosis, Mastery Learning & Remediation," Proc. Of 7.sup.th World Conf. On Artificial Intelligence in Education, pp. 123-130 Aug. 1995.
J. Reye, "A Goal-Centered Architecture for Intelligent Tutoring Systems," Proc. Of 7.sup.th World Conf. On Artificial Intelligence in Education, pp. 307-314 Aug. 1995.
J. Siemer and M.C. Angelides, "Evaluating Intelligent Tutoring with Gaming-Simulations," Proc. Of the 1995 Winter Simulation Conf., pp. 1376-1383 Dec. 1995.
S.J.E. Taylor and J. Siemer, Enhanced Simulation Education with Intelligent Tutoring Systems, Proc. Of the 1996 Winter Simulation Conf., pp. 675-680 Dec. 1996.

K. Nakabayashi, et al., "Architecture of an Intelligent Tutoring System on the WWW," Proc. Of 1997 World Conf. On Artificial Intelligence in Education, pp. 39-46 Dec. 1997.
W. van Joolingen, et al., "The SimQuest Authoring System for Simulation-Based Discovery Learning," Proc. Of 1997 World Conf. On Artificial Intelligence in Education, pp. 79-86 Dec. 1997.
M. Papagni, et al., "Teaching through Case-Based Reasoning: An ITS Engine Applied to Business Communication", Proc. Of 1997 World Conf. On Artificial Intelligence in Education, pp. 111-118 Dec. 1997.
R. Azevedo, et al., "RadTutor: The Theoretical and Empirical Basis Basis for the Design of a mammography Interpretation Tutor," Proc. Of 1997 World Conf. On

Artificial in Education, pp. 386-393.R.H. Kemp, "Using the Wizard of Oz Technique to Prot" Dec. 1997.
R.H. Kemp, "Using the Wizard of Oz Technique to Prototype a Scenario-Based simulation Tutor," Proc. Of 1997 World Conf. On Artificial Intelligence in Education, pp. 458-465 Dec. 1997.
P. Busilovsky, et al., "Distributed Intelligent Tutoring on the Web," Proc. Of 1997 World Conf. Of Artificial Intelligence in Education, pp. 482-489 Dec. 1997.
C.D. Robinson, et al., "Briding the Virtual and the Physical: The InterSim as a Collaborative Support Interface," Proc of 1997 World Conf. On Artificial Intelligence in Education, pp. 556-558 Dec. 1997.
K. Itoh, et al., "An Object-Oriented Architecture for Evolutional Development of Interactive Learning Environment with Coached Problem-Solving," Proc. Of 1997 World Conf. On artificial Intelligence in Education, pp. 592-594 Dec. 1997.
B.T. Cheok and A.Y.C. Nee, "Developing a Design System into an Intelligent Tutoring System," Int'l. J. Engr. Eud., vol. 13(5), pp. 341-346 Dec. 1997.
J. Breuker, "What are Intelligent Coaching Systems and Why are They (in)evitable?" IEE Colloquium on Artificial Intelligence in Educational Software, pp. 2/1-2/5 Jun. 1998.
J.S. Brown, et al., "Pedagogical, natural language and knowledge engineering techniques in Sophie I, II and III," in Intelligent Tutoring Systems, D. Sleeman and J.S. Brown eds., pp. 227-282 Dec. 1982.
S.S. Prabhu and A. Srivastava, "Computer Aided Instruction for Statistics: A Knowledge-Based Systems Approach," Int'l J. of Computers in Adult Education and Training, vol. 5(1), pp. 3-14.

ART-UNIT: 272

PRIMARY-EXAMINER: Hafiz; Tariq R.

ASSISTANT-EXAMINER: Starks; Wilbert

ATTY-AGENT-FIRM: Hickman Stephens & Coleman, LLP

ABSTRACT:

A system is disclosed that provides a goal based learning system utilizing a rule based expert training system to provide a cognitive educational experience. The system provides the user with a simulated environment that presents a business opportunity to understand and solve optimally. Mistakes are noted and remedial educational material presented dynamically to build the necessary skills that a user requires for success in the business endeavor. The system utilizes an artificial intelligence engine driving individualized and dynamic feedback with synchronized video and graphics used to simulate real-world environment and interactions. Multiple "correct" answers are integrated into the learning system to allow individualized learning experiences in which navigation through the system is at a pace controlled by the learner. A robust business model provides support for realistic activities and allows a user to experience real world consequences for their actions and decisions and entails realtime decision-making and synthesis of the educational material. The system is architected around a linked list activity table utilized to manage and control the system.

19 Claims, 76 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L21: Entry 17 of 21 File: USPT Oct 17, 2000

US-PAT-NO: 6134540
DOCUMENT-IDENTIFIER: US 6134540 A

TITLE: System, method, and program for applying query rewrite technology to object building

DATE-ISSUED: October 17, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Carey; Michael J.	San Jose	CA		
Kiernan; Gerald G.	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 853294 [PALM]
DATE FILED: May 9, 1997

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/2; 707/3, 707/4
US-CL-CURRENT: 707/2; 707/3, 707/4

FIELD-OF-SEARCH: 707/4, 707/2, 707/3, 704/4

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5295256	March 1994	Bapat	707/3
☐	5548755	August 1996	Leung et al.	707/4
☐	5764973	June 1998	Lunceford et al.	707/1
☐	5765147	June 1998	Mattos et al.	707/4
☐	5774692	June 1998	Boyer et al.	709/300
☐	5778355	July 1998	Boyer et al.	707/2
	5797136	August 1998	Boyer et al.	707/2

☐				
☐	5809505	September 1998	Lo et al.	707/102
☐	5835757	November 1998	Qulid et al.	707/10
☐	5864840	January 1999	Leung et al.	707/2
☐	5930785	July 1999	Lohman et al.	707/2
☐	5991754	November 1999	Raitto et al.	707/2

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 320 266 A2	June 1989	EP	
0 702 312 A1	March 1996	EP	
WO 95/01610	January 1995	WO	

OTHER PUBLICATIONS

Gio Wiederhold, "Views, Objects, and Databases," 8153 Computer, No. 12, New York, NY, pp. 37-44, Dec. 19, 1986.
Chih-Chin Liu, et al., "Object View Derivation and Object Query Transformation," Proc. Eighteenth Annual Int. Computer Software and Application Conference, Taipei, Taiwan, pp. 157-162, Nov. 9-11, 1994.
"Updating Relational Databases through Object-Based Views"; Thierry Barsalou, Arthur M. Keller, Niki Siambela, Gio Wiederhold; Proc. ACM-SIGMOD International Conference on Management of Data, Denver, Jun. 1991.
"The MultiView OODB View System: Design and Implementation;" Harumi A. Kuno and Elke A. Rundensteiner; University of Michigan Technical Report CSE-TR-241-95.
"Object Views: Extending the Vision"; Sandra Heiler, Stanley Zdonik; Proc. IEEE International Conference on Data Engineering 90, Apr. 1990.
"Objects and Views"; Serge Abiteboul, Anthony Bonner; ACM-SIGMOD International Conference on Management of Data, ACM Feb. 1991.
"On View Support in Object-Oriented Database Systems"; Won Kim, William Kelley; Modern Database Systems: The Object Model, Interoperability, and Beyond, Part 1/Next-Generation Database Technology, chapter 6, 1995.
"Towards Heterogeneous Multimedia Information Systems: The Garlic Approach"; M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams and E. L. Wimmers; Proc. 1995, IEEE Workshop on Research Issues in Data Engineering, Taipei, Taiwan, Mar. 1995.
"Queries and Views in an Object-Oriented Data Model;" U. Dayal; Proc. 2nd International Workshop on Database Programming Languages; editors, Richard Hull, Ron Morrison, and David Stemple, Gleneden Beach, Jun. 1989.
Third Generation Data Base System Manifesto, Mike Stonebraker et al, Computer Standards & Interfaces, 12, Dec. 1991.
"Object-Oriented Database Systems: Promise, Reality, and Future," Won Kim, Proc. 19th International Conference on Very Large Data Bases, Dublin, Aug. 1993.
"A Data Model and Query Language for EXODUS," Proc. ACM-SIGMOD International Conference on Management of Data, Carey, Michael; DeWitt, David; Vandenberg, Scott; Chicago, Jun. 1988.
"A Model of Queries for Object-Oriented Databases," Kim, Won; Proc. 15th International Conference on Very Large Data Basses, Amsterdam, Aug. 1989.
"A Query Language for the O.sub.2 Object-Oriented Database System," Bancilhon, Francois; Cluet, S.; Delobel, C.; Proc. 2.sup.nd International Workshop on Database Programming Languages, Hull, Richard; Morrison, Ron; Stemple, David, editors;

Gleneden Beach, Jun. 1989, Morgan-Kaufmann Publishers, Inc.
"Query Processing in the ObjectStore Database System," Orenstein, Jack; Haradhvala, Sam; Margulies, Benson; Sakahara, Don; Proc. International Conference on Management of Data, San Diego, Jun. 1992.
"CQL++: A SQL for a C++ Based Object-Oriented DBMS," Dar. S.; Gehani, N.; Jagadish, H.; Proc. International Conference on Extending Data Base Technology, Advances in Database Technology--EDBT '92. Lecture Notes in Computer Science, Vienna, 1992. Springer-Verlag.
"Querying Object-Oriented Databases," Kifer, Michael; Kim, Won; Sagiv, Yehoshua; Proc. ACM-SIGMOD International Conference on Management of Data, San Diego, Jun. 1992.
"Object Query Language," Atwood, Tom; Duhl, Joshua; Ferran, Guy; Loomis, Mary; Wade, Drew; Object Database Standards: ODMG--93 Release 1.1, R.G.G. Cattell, editor, Morgan-Kaufmann Publishers, Inc., 1993.
"Experiences building the open oodb query optimizer," Blakeley, Jose; McKenna, William J.; Graefe, Goetz, Proc. ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 1993.
"Observations on the ODMG-93 Proposal," Kim, W., ACM SIGMOD Record, 23(1), Mar. 1994.
"Enabling the Integration of Object Applications with Relational Databases"; Persistence Software, Inc.; http://www.persistence.com/persistence/pageTwo.pages/techoview.htn; Apr. 2, 1997 1:40PM.
"Extensible/Rule Based Query Rewrite Optimization in Starburst," Hamid Pirahesh, Joseph M. Hellerstein, and Wagar Hasan, In Proc. ACM-SIGMOD International Conference on Management of Data, San Diego, Jun. 1992.
"Magic is Relevant," Inderpal Singh Mumick, Sheldon J. Finkelstein, Hamid Pirahesh, and Raghu Ramakrishnan, In Proc. ACM-SIGMOD International Conference on Management of Data, pp. 247-258, Atlantic City, May 1990.
"The Magic of Duplicates and Aggregates," Inderpal Singh Mumick, Hamid Pirahesh, and Raghu Ramakrishnan, In Proc. 16.sup.th International Conference on Very Large Data Bases, Brisbane, Aug. 1990.
"A General Framework for the Optimization of Object-Oriented Queries," Sophie Cluet and Claude Delobel, In Proc. ACM-SIGMOD International Conference on Management of Data, San Diego, Jun. 1992.
"OMG. Object Services Request for Proposals," OMG TC Document 94.4.18, 1994.
"OMG. Object Query Service Specification, Joint Submission," OMG TC Document 95.1.1, 1995) .
ISO/IEC 9075:1992, Database Language SQL.
Microsoft. Programmer's Reference, Microsoft Open Database Connectivity Software Development Kit, 1992.).
Kiernan et al., "Extending SQL-92 for OODB Access: Design and Implementation Experience", ACM 0-89791, p. 467-480, Jan. 1995.

ART-UNIT: 271

PRIMARY-EXAMINER: Alam; Hosain T.

ASSISTANT-EXAMINER: Corrielus; Jean M.

ATTY-AGENT-FIRM: Pretty, Schroeder & Poplawski

ABSTRACT:

The system, method, and program of this invention enables an object language application (e.g., C++, JAVA, etc.,) to issue a query over a view and to receive back, as query results, handles to application type objects which can be further manipulated by the application. A view is defined herein as a collection of a view type, and a view type is defined as a class or type. Upon receipt of the query referencing a view type, a query engine generates a query plan that builds mock

(i.e., proxy) application type objects in memory based upon the view types. The application can run methods on the application type objects or point to other application type objects from the handles, to the application objects, that are returned to the application; and these manipulations will be understood by the query engine. In a preferred embodiment, query rewrite optimizations are applied to the queries over views requiring object building in order to optimize the evaluation of the query and the building of view objects as query results. For example, when a query over a view is analyzed and it is determined that the query is not requesting a handle, and is not referencing a method, but only asks for values, no objects are built. Also, if a query traverses a reference type attribute, but the query can be transformed into a join or outer join operation between relational tables, then no object building is required. In these above described situations, the rewritten query can be pushed down to the database management system of the data source for resolution. If the query does request a handle or references a method, then some object building is required. However, query rewrite techniques can still be applied so that parts of the query are pushed down to the DBMS to minimize the number of objects that are built.

31 Claims, 11 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L21: Entry 20 of 21 File: USPT Jun 29, 1999

US-PAT-NO: 5918232
DOCUMENT-IDENTIFIER: US 5918232 A

TITLE: Multidimensional domain modeling method and system

DATE-ISSUED: June 29, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Pouschine; Nicholas	Fremont	CA		
Stross; Kenner G.	Oakland	CA		
Brill; Michael L.	San Francisco	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Whitelight Systems, Inc.	Palo Alto	CA			02

APPL-NO: 08/ 978168 [PALM]
DATE FILED: November 26, 1997

INT-CL: [06] G06 F 17/30

US-CL-ISSUED: 707/103; 707/2, 707/3, 707/4
US-CL-CURRENT: 707/103R; 707/2, 707/3, 707/4

FIELD-OF-SEARCH: 707/103, 707/2, 707/3, 707/4

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5367619	November 1994	Dipaolo et al.	395/149
☐	5495608	February 1996	Antoshenkov	
☐	5560007	September 1996	Thai	
☐	5592666	January 1997	Perez	
☐	5664172	September 1997	Antoshenkov	
☐	5666528	September 1997	Thai	
☐	5742738	April 1998	Koza et al.	395/13

ART-UNIT: 271

PRIMARY-EXAMINER: Amsbury; Wayne

ASSISTANT-EXAMINER: Lewis; Cheryl R.

ATTY-AGENT-FIRM: Guernsey; Larry B. Hughes; Michael J.

ABSTRACT:

A system and method for computer modeling (10) and for creating hyperstructures (51) which are to be contained in a computer memory, which obtains measurements of physical objects and activities which are related to the entity to be modeled in the computer hyperstructure (51). The measurements are transformed into computer data which corresponds to the physical objects and activities external to the computer system (10). A plurality of independent dimensions (54) are created, where each dimension (54) includes at least one element (58). A plurality of cells (56) are created, each of which is associated with the intersection of two or more elements (58), each cell (56) being capable of storing at least one value. At least one rule domain (60) is associated with at least one cell (56), the rule domain (60) including at least one rule for assigning values to the associated cells (56). A domain modeling rule set (126) is prepared (300), which determines which of the rules will provide the value associated with each of the cells (56) wherein application of the domain modeling rule set (126) to the hyperstructure (51) causes a physical transformation of the data corresponding to said physical objects which are modeled in said hyperstructure (51).

Also disclosed is a method for querying computer hyperstructures (51), a Hyperstructure Query Language, and a "cell explorer", which allows direct viewing of the applied formulas that produce a specific value for a cell (56).

18 Claims, 17 Drawing figures

First Hit Fwd Refs

End of Result Set

☐ Generate Collection Print

L21: Entry 21 of 21 File: USPT Jan 19, 1999

US-PAT-NO: 5862325
DOCUMENT-IDENTIFIER: US 5862325 A

TITLE: Computer-based communication system and method using metadata defining a control structure

DATE-ISSUED: January 19, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Reed; Drummond Shattuck	Seattle	WA		
Heymann; Peter Earnshaw	Seattle	WA		
Mushero; Steven Mark	Seattle	WA		
Jones; Kevin Benard	Seattle	WA		
Oberlander; Jeffrey Todd	Seattle	WA		
Banay; Dan	Seattle	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Intermind Corporation	Seattle	WA			02

APPL-NO: 08/ 722314 [PALM]
DATE FILED: September 27, 1996

PARENT-CASE:
This application is a continuation-in-part of co-pending application Ser. No. 08/609,115, filed Feb. 29, 1996.

INT-CL: [06] G06 F 17/30, G06 F 17/40

US-CL-ISSUED: 395/200.31; 395/200.42, 395/200.58, 395/200.72, 395/200.74, 707/10, 707/203, 707/204
US-CL-CURRENT: 709/201; 704/270.1, 707/10, 707/203, 707/204, 709/212, 709/228, 709/242, 709/244

FIELD-OF-SEARCH: 395/200.3-200.33, 395/200.42, 395/200.46-200.49, 395/200.57-200.59, 395/200.62, 395/200.72-200.74, 395/702-70, 707/200-204, 707/100-103, 707/10

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4274139	June 1981	Hodgkinson et al.	395/200.33
☐	4432057	February 1984	Daniell et al.	707/8
☐	4558413	December 1985	Schmidt et al.	707/203
☐	4604686	August 1986	Reiter et al.	395/500
☐	4714992	December 1987	Gladney et al.	707/206
☐	4714995	December 1987	Materna et al.	707/201
☐	4745559	May 1988	Willis et al.	705/37
☐	4746559	May 1988	Nishikawa	428/142
☐	4815030	March 1989	Cross et al.	707/10
☐	4974149	November 1990	Valenti	345/200.47
☐	5008814	April 1991	Mathur	395/200.51
☐	5008853	April 1991	Bly et al.	345/331
☐	5019963	May 1991	Alderson et al.	707/201
☐	5133075	July 1992	Risch	707/201
☐	5155847	October 1992	Kirouac et al.	395/200.51
☐	5187787	February 1993	Skeen et al.	395/680
☐	5220657	June 1993	Bly et al.	711/152
☐	5226161	July 1993	Khovi et al.	395/683
☐	5257369	October 1993	Skeen et al.	395/680
☐	5287504	February 1994	Carpenter et al.	707/201
☐	5303379	April 1994	Khovi et al.	395/710
☐	5359730	October 1994	Marron	395/712
☐	5404488	April 1995	Kerrigan et al.	711/133
☐	5426747	June 1995	Weinreb et al.	711/203
☐	5440744	August 1995	Jacobson et al.	395/200.33
☐	5452447	September 1995	Nelson et al.	707/205
☐	5473772	December 1995	Halliwell et al.	395/712
☐	5485370	January 1996	Moss et al.	395/200.47
☐	5491820	February 1996	Belove et al.	707/3
☐	5495610	February 1996	Shing et al.	395/200.51
☐	5497491	March 1996	Mitchell et al.	395/683
☐	5499343	March 1996	Pettus	395/200.33
☐	5515508	May 1996	Pettus et al.	395/200.33
☐	5519769	May 1996	Weinberger et al.	379/112
☐	5519875	May 1996	Yokoyama et al.	395/683
☐	5528490	June 1996	Hill	395/712

☐	5548726	August 1996	Pettus	395/200.51
☐	5555427	September 1996	Aoe et al.	395/200.31
☐	5557793	September 1996	Koeber	707/103
☐	5560012	September 1996	Ryu et al.	395/701
☐	5564051	October 1996	Halliwell et al.	707/200
☐	5566302	October 1996	Khalidi et al.	395/200.31
☐	5577244	November 1996	Killebrew et al.	395/703
☐	5581755	December 1996	Koeber et al.	707/103
☐	5581761	December 1996	Radia et al.	395/702
☐	5581764	December 1996	Fitzgerald et al.	395/703
☐	5586311	December 1996	Davies et al.	707/1
☐	5586326	December 1996	Ryu et al.	395/701
☐	5596720	January 1997	Hamada et al.	395/200.36
☐	5596746	January 1997	Shen et al.	707/101
☐	5600834	February 1997	Howard	707/201
☐	5608874	March 1997	Ogawa et al.	395/200.76
☐	5615112	March 1997	Liu Sheng et al.	707/104
☐	5619710	April 1997	Travis, Jr. et al.	395/200.33
☐	5623656	April 1997	Lyons	707/10
☐	5623661	April 1997	Hon	707/1
☐	5625818	April 1997	Zarmer et al.	707/104
☐	5630092	May 1997	Carreiro et al.	711/111
☐	5630116	May 1997	Takaya et al.	707/201
☐	5634010	May 1997	Ciscon et al.	395/200.33
☐	5640564	June 1997	Hamilton et al.	395/683
☐	5644764	July 1997	Johnson et al.	707/103
☐	5649192	July 1997	Stucky	707/103
☐	5652887	July 1997	Dewey et al.	395/683
☐	5668997	September 1997	Lynch-Freshner et al.	395/683
☐	5673322	September 1997	Pepe et al.	380/49
☐	5682532	October 1997	Remington et al.	395/683
☐	5684984	November 1997	Jones et al.	707/10
☐	5684991	November 1997	Malcolm	707/204
☐	5689708	November 1997	Regnier et al.	395/200.59
☐	5706434	January 1998	Kremen et al.	395/200.48
☐	5710918	January 1998	Lagarde et al.	707/10
☐	5721911	February 1998	Ha et al.	707/100
	5761677	June 1998	Senator et al.	707/203

☐

☐ 5761678 June 1998 Bendert et al. 707/204

OTHER PUBLICATIONS

C. Bowman, P. Danzig, D. Hardy, U. Manber, M. Schwartz & D. Wessels "Harvest: A Scalable, Customizable Discovery and Access System"Mar. 12, 1995.
D. Hardy & M. Schwartz "Customized Information Extraction as a Basis for Resource Discovery"Mar. 1994.
William G. Camargo "The Harvest Broker, " Dec. 1994.
D. Bulterman, G. van Rossum and R. van Liere "A Structure for Transportable, Dynamic Multimedia Documents"USENIX, Summer '91 Nashville, TN.
G. Almes and C. Holman "Edmas: An Object-Oriented, Locally Distributed Mail System "IEEE Transctions on Software Engineering, Sep. 1987.
G. Almes, A. Black, C. Bunie and Wiebe "Edmas: A Locally Distributed Mail System"IEEE, 1984.
W. Bender, H. Lie, J. Orwant, L. Teodosio, & N. Abramson "Newspace: Mass Media and Personal Computing," USENIX-Summer '91 -Nashville TN.
R. Thomas, H. Forsdick, T. Crowley, R. Schaaff, R. Tomlinson & V. Travers "Diamond: A Multimedia Message System Built on a Distributed Architecture"IEEE, Dec. 1994.
S. Ramanathan & P. V. Rangan "Architectures for Personalized Multimedia"IEEE, 1994.

N. Yankelovich, B. Haan, N. Meyrowitz & S. Drucker "Intermedia: The Concept and the Construction of a Seamless Information Environment"IEEE, Jan. 1988.
D. Woelk, W. Kim & W. Luther "An Object-Oriented Approach to Multimedia Database"ACM 1986.
N. Borenstein, C. Everhart, J. Rosenberg, A. Stoller "A Multi-media Message System for Andrew"USENIX Winter Conference Feb., 1988.
S. Jackson & N. Yankelovich "InterMail: A Prototype Hypermedia Mail System"Hypertext 91 Proceedings Dec. 1991.
E. Hoffert & G. Gretsch, "The Digital News System at Ed.ucom: A Convergence of Interactive Computing Newspaper, Television and High Speed Networks"Communications of the ACM Apr. 1991.
D. Crocker, E. Szurkowski & D. Farber "An Internetwork Memo Distribution Capability--MMDF"IEEE, ACM 1979.
Douglas Engelbart "Authorship Provisions in Augment"IEEE, 1984.
J.J. Garcia-Luna-Aceves "Towards Computer-Based Multimedia Information Systems"Computer Message System 85, 1986.
Debra P. Deutsch "Implementing Distribution Lists in Computer-Based Message Systems"Computer-Based Message Services, IFIP, 1984.
T. Purdy, D. Thorslund & N. Witchlow "Meridian SL Messaging"Computer Message Systems-85 IFIP, 1986.
Michael Tschichholz "Message Handling System: Requirements to the User Agent"Computer Message Systems-85, IFIP, 1986.
Lother Wosnitza "Group Communication in the MHS Context"Computer Message Systems 85 IFIP, 1986.
Jacob Palme "Distribution Agents (mailing lists) in Message Handling Systems"Computer Message Systems 85 IFIP, 1986.
Teresa F. Lunt "A Model for Message System Security"Computer Message Systems 85 IFIP, 1986.
A. Roger Kaye "A User Agent for Multiple Computer-Based Message Services"Computer-Based Message Services, IFIP 1984.
Paul Wilson "Structure for Mailbox System Applications"Computer-Based Message Services, IFIP 1984.
J. Postel, G. Finn, A. Katz & J. Reynolds "The ISI Experimental Multimedia Mail System"Information Sciences Institute, Sep. 1986.
E. Moeller, A. Scheller & G. Schurmann "Distributed Processing of Multimedia Information"IEEE Computer Society Proceedings May 28-Jun. 1, 1990.

Richard L. Phillips "An Interpersonal Multimedia Visualization System"IEEE Computer Graphics & Applications IEEE 1991.
Jacob Palme "You Have 134 Unread Mail! Do You Want to Read Them Now?" Computer-Based Message Services IFIP, 1984.
Michael Caplinger "An Information System Bsed on Distributed Objects"OOPSLA '87 Proceedings.
M. Papa, G. Raguccini, G. Corrente, M. Ferrise, S. Giurleo and D. Vitale "The Development of an Object-Oriented Multimedia Information System"Lecture Notes in Computer Science, Sep. 1994.
Silvano Maffeis "A Flexible System Design to Support Object-Groups and Object-Oriented Distributed Programming"Lecture Notes in Computer Science, Jul. 1993.
R. Gotze, H. Eirund & R. Claass en "Object-Oriented Dialog Control for Multimedia User Interfaces"Lecture Notes in Computer Science-Human Computer Interaction Sep. 1993.
Chris Maeda "A Metaobject Protocol for Controlling File Cache Management"Lecture Notes in Computer Science, Mar. 1996.
A. Joseph, A. deLespinasse, J. Tauber, D. Gifford & M. Kaashoek "Rover" A Toolkit for Mobile Information Access SIGOPS '95 1995. ACM
Wolfgang Lux "Adaptable Object Migration: Concept and Implemenetaion"Operating Systems Review Apr. 1995.
R. Campbell, N. Islam, R. Johnson, P. Kougiouris & P. Madany "Choices, Frameworks and Refinement" Deptment of Computer Science, University of Illinois, Dec. 1991.
Klemens Bohm & Thomas C. Rakow "Metadata for Multimedia Documents"SIGMOD Record, Vol. 23, No. 4, Dec. 1994.
Simon Gibbs "Compostie Multimedia and Active Objects"OOPSLA '91.
T. Purdin, R. Schlichting & G. Andrews "A File Replication Facility for Berkeley Unix"Software Practive and Experience, Vol. 17, Dec. 1987.
A. Black, N. Hutchinson, E. Jul & H. Levy "Object Structure in the Emerald System"OOPSLA '86 Proceedings.
Daniel T. Chang "Coral: A Concurrent Object-Oriented System for Constructing and Executing Sequential, Parallel and Distributed Applications"OOPS Messenger, Apr. 1991.
A. Birrell, G. Nelson, S. Owicki & E. Wobber "Network Objects"Proceedings of the 14th ACM Symposium on Operating Systems Principles, Dec. 5-8, 1993.
Jacques Ferber "Computational Reflection in Class based Object Oriented Languages"OOPSLA '89 Proceedings.
Michael Caplinger "An Information System Based on Distributed Objects"OOPSLA '87 Proceedings.
C. Fung & M. Pong "MOCS: an Object-Oriented Programming Model for Multimedia Object Communication and Synchronization"1994 IEEE.
T. Hase & M. Matsuda "A New Audio-Visual Control Using Message Object Transmission", 1994 IEEE, Nov. 1994.
F. Horn & J. Stefani "On Programming and Supporting Multimedia Object Synchronization" The Computer Journal, Vol. 36, No. 1, 1993.
T. Little & A. Ghafoor Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks, IEEE, 1991.
M. Vazirgiannis & C. Mourlas "An Object-Oriented Model for Interactive Multimedia Presentations"The Computer Journal, Vol. 36, No. 1, 1993.
T. Little & A. Ghafoor "Synchronization and Storage Models for Multimedia Objects" 1990 IEEE, Apr. 1990.
Cosmos Nicolaou "Architecture for Real-Time Multimedia Communications Systems", 1990 IEEE, Apr. 1990.
Ralf Steinmetz "Synchronization Properties in Multimedia Systems" 1990 IEEE, Apr. 1990.
T. Little & A. Ghafoor "Network Considerations for Distributed Multimedia Object Composition and Communication" 1990 IEEE Network Magazine, Nov. 1990.
K. Smith and S. Zdonik "Intermedia: A Case Study of the Differences Between Relational and Object-Oriented Database Systems" OOPSLA '87 Proceedings.
N. Yankelovich, B. Haan, N. Meyrowitz & S. Drucker "Intermedia: The Concept and the Construction of a Seamless Information Environment" Jan. 1988 IEEE.

S. Ramanathan & P. Rangan "Architectures for Personalized Multimedia "1994 IEEE.
Marvin Sirbu and J. D. Tygar, "Netbill: An Internet Commerce system Optimized For Network-Delivered Services", IEEE Personal Communications Magazine, pp. 34-39, Aug. 1995.
Henrik Eriksson, "Expert System As Knowledge Servers", IEEE Expert Magazine, pp. 14-19, Jun. 1996.
Budi Yuwono and Dik Lun Lee, "Wise: A World Wide Web Resource Database System", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. Aug. 1996.
H. Penny Nii "Blackboard Systems"The Al Magazine, Summer, 1986.
AppleShare, Apr. 1995.
"Manual Page for Unix NFS Mount Command".
"Manual Page for Unix FSTAB Command".
Phil Lapsley and Brian Kantor "Network News Transfer Protocol", Feb. 1986.
Brian Kantor and Phil Lapsley, Network News Transfer Protocol, "A Proposed Standard for the Stream-Based Transmission of News", Feb. 1986.
M. Crispin "Network Working Group", University of Washington, Dec. 1996.
Terry Gray Comparing Two Approaches to Remote Mailbox Access: IMAP vs. POP, University of Washington.
Terry Gray "Message Access Paradigms and Protocols", University of Washington, Aug. 1995.

ART-UNIT: 278

PRIMARY-EXAMINER: Lim; Krisna

ASSISTANT-EXAMINER: Barot; Bharat

ATTY-AGENT-FIRM: Wolf, Greenfield & Sacks, PC

ABSTRACT:

An automated communications system operates to transfer data, metadata and methods from a provider computer to a consumer computer through a communications network. The transferred information controls the communications relationship, including responses by the consumer computer, updating of information, and processes for future communications. Information which changes in the provider computer is automatically updated in the consumer computer through the communications system in order to maintain continuity of the relationship. Transfer of metadata and methods permits intelligent processing of information by the consumer computer and combined control by the provider and consumer of the types and content of information subsequently transferred. Object oriented processing is used for storage and transfer of information. The use of metadata and methods further allows for automating may of the actions underlying the communications, including communication acknowledgements and archiving of information. Service objects and partner servers provide specialized data, metadata, and methods to providers and consumers to automate many common communications services and transactions useful to both providers and consumers. A combination of the provider and consumer programs and databases allows for additional functionality, including coordination of multiple users for a single database.

126 Claims, 57 Drawing figures

L21: Entry 21 of 21 File: USPT Jan 19, 1999

DOCUMENT-IDENTIFIER: US 5862325 A
TITLE: Computer-based communication system and method using metadata defining a control structure

Abstract Text (1):
An automated communications system operates to transfer data, metadata and methods from a provider computer to a consumer computer through a communications network. The transferred information controls the communications relationship, including responses by the consumer computer, updating of information, and processes for future communications. Information which changes in the provider computer is automatically updated in the consumer computer through the communications system in order to maintain continuity of the relationship. Transfer of metadata and methods permits intelligent processing of information by the consumer computer and combined control by the provider and consumer of the types and content of information subsequently transferred. Object oriented processing is used for storage and transfer of information. The use of metadata and methods further allows for automating may of the actions underlying the communications, including communication acknowledgements and archiving of information. Service objects and partner servers provide specialized data, metadata, and methods to providers and consumers to automate many common communications services and transactions useful to both providers and consumers. A combination of the provider and consumer programs and databases allows for additional functionality, including coordination of multiple users for a single database.

Brief Summary Text (3):
The present invention relates to data communications systems. More particularly, it relates to an automated communications system which coordinates the transfer of data, metadata, and instructions between databases in order to control and process communications.

Brief Summary Text (6):
Establishing, maintaining, operating, and even terminating any one of these types of communications relationships involves significant work on the part of both the provider and consumer. For example, to initiate any type of communications relationship, providers must first locate the consumers with whom to communicate and vice versa. Solving this problem is subject of several entire industries, such as the directory industry, the mailing list industry, and the advertising industry. Once a provider or consumer has been identified, contact information (e.g., names, titles, addresses, telephone numbers, electronic mail addresses, etc.) must be exchanged between the provider and consumer. This contact information must be maintained by both parties so that future communications can be effected as needed. When the contact information changes for an entity, all providers or consumers having relationships with the entity must be notified of the changes, who in turn must update their own records. This work also extends to other data and records exchanged in the context of the communications relationship, e.g. orders, receipts, product numbers, invoice numbers, customer numbers, notes, brochures, reports, etc. Maintenance of this information requires significant human time involvement for receiving information, storing information, indexing information, searching for

desired information, and retrieving information. The human component of record maintenance also creates a potential for error, which can cause the information to be faulty or to become lost.

Brief Summary Text (23):
Another approach to automating communications and data transfers is shared replicated database systems such as Lotus Notes and Collabra Share. With these systems, information to be communicated is entered via a client program into one or more databases which may reside locally on client computers or on network server computers. These databases are then replicated to other server computers or local client computers throughout the system so that the data can be easily accessed by any other user of the system who needs the information and has the proper access privileges. Access privileges are controlled by one or more system administrators via the system servers. Some of these systems, notably Collabra Share, also allow users to "subscribe" to specific databases. These users can receive an e-mail notification from a database agent monitoring the database when a new entry or a certain condition has been made in that database. These systems may also employ electronic forms and forms processing languages to structure the data being entered into a database, and to take programmable actions based on the data entered. The architecture of these systems is designed for groups of users to share information related to specific topics, and to automate the transfer of data between different computer applications used by an organization. For this reason the core data structure of the architecture is a subject database or "forum". Each subject database covers a number of related interest topics under which all entries in the database are categorized. All copies of any subject database are synchronized throughout the system when data in any one copy has been changed.

Brief Summary Text (24):
While suitable for information sharing amongst the members of a group, this architecture is not well suited for automating communications relationships among a large number of information providers and consumers. First, all the providers and consumers need to be interconnected through the system in order to communicate. This could be done by having all providers and consumers enroll in one large system in which they all had access privileges. In such a system each provider would need to have at least one subject database for communicating with his/her consumers. This enormous number of subject databases would then need to be replicated among the large number of servers required to service the complete population of the system, which would quickly overwhelm the capacity of the servers or network to handle replication. A more realistic alternative would be to have each provider or group of providers operate and administer their own system, making their internal subject databases available to consumers via public data networks such as the Internet. Consumers would use the system client software to "subscribe" to the subject databases of each provider with which they desire to communicate. Only the subject databases a consumer subscribed to would be replicated on his/her desktop. This solution would spread the replication load to a large number of servers, each handling a smaller amount of traffic. However, each server would now have to manage replication for a large number of external consumers as well as internal group members. There is no easy way to distribute this replication load to the consumer's computer. Second, subject databases do not allow the consumer to control and filter the incoming communications from providers. Consumers must still scan the databases for items of interest. Providers could overcome this by creating a subject database for each interest topic, but the additional administrative and server replication overhead would strongly discourage this. Third, because notification of new information is handled via a separate application, e-mail, the consumer is forced to coordinate notification and data storage/response among two communications systems. Fourth, since subject databases are replicated from the servers, they do not give consumers an easy way to copy or transfer them to other consumers. Finally, because the entire system depends on server-based replication, administrative changes or reconfigurations of these servers such as system name or address changes require administrative updates to all subscribing consumers, a job

which consumers must handle manually.

Brief Summary Text (32):
According to another aspect of the invention, a provider program is used to create, edit, and maintain data, metadata and instructions in a provider database. The provider program also controls distribution of the information to various consumers. Different information contained in the provider database can be transferred and used in communications relationships with different consumers. The provider database includes information associating the information with each potential recipient. The association information is used to selectively distribute information and information updates. The provider program also receives and uses information from the consumer computer to control encoding and transfer of information to the consumer computer. According to another aspect, the provider program uses a markup language to format the information for transfer.

Brief Summary Text (33):
According to another aspect of the invention, a consumer program is used to receive and process the transferred information. The consumer program receives information from the provider or polls a location identified by the transferred information to determine when information has been updated by the provider. The consumer program then retrieves the information from the proper source and compares it to the existing information to determine what has been updated. The consumer program maintains a database of information from different providers. When updated information is received, the consumer program executes instructions associated with the information to store the updated information, notify a user of updated information, and generate responses for the consumer. The consumer program also can transfer the information to second consumer computer. The second consumer computer can obtain updated information from the provider computer or have it forwarded by the first consumer computer.

Brief Summary Text (36):
According to another aspect of the invention, the provider and consumer programs and databases are combined to obtain additional functionality. The communication system can allow multiple users for a single program and database. The data, metadata, and instructions coordinate the operation of the programs for each user and allow for communications between users of the single database.

Drawing Description Text (7):
FIGS. 6A and 6B illustrate object oriented data structures for storing system ID data within the system ID database and for user objects.

Drawing Description Text (19):
FIG. 17 illustrates the object oriented database structures for different communications object types.

Drawing Description Text (31):
FIGS. 29A and 29B illustrate object oriented data structures for a directory system and a message or discussion thread.

Drawing Description Text (32):
FIG. 30 is a block flow diagram for a process for creating directory listings using service objects and partner servers.

Drawing Description Text (33):
FIGS. 31A and 31B are block flow diagrams of processes for updating directory listings and monitoring category objects using service objects and partner servers.

Detailed Description Text (3):
There is illustrated in FIG. 1 a first embodiment of a system of the present

invention which automatically updates a database in a consumer computer with information from a provider computer. Numerous providers and consumers exist in the system of the present invention. However, since all communications can be separated into transfers between a single provider and consumer, the design and operation of the system is illustrated with only one provider and one consumer, except as otherwise noted. As illustrated in FIG. 1, a provider computer 1 includes a provider database 11 of communications control information which it desires to disseminate or make accessible to one or more consumers. A consumer computer 2 includes a consumer database 21 of communications control information received from providers and stored by the consumer. The organization, structure, and content of the provider database 11 and consumer database 21 are discussed below. The provider computer 1 is connected through a communications network 3 to the consumer computer 2. Any communications network 3 may be used to connect the provider computer 1 and the consumer computer 2, including direct network connections, server-based environments, telephone networks, the Internet, intranets, local area networks (LANS), wide area networks (WANS), the World Wide Web, other webs, and even transfers of data on physical media such as disks or computer-readable paper outputs via postal communications networks. The particulars of the communications network illustrated as preferred embodiments are not limiting features of the invention. However, the Internet and World Wide Web provide existing capabilities between computers sufficient to provide the necessary connections. For this reason, the description of the present invention is based on this communications medium, which should be understood to be used for purpose of illustration only. Organization and operation of the Internet and communications over the Internet are discussed generally in Kris Jamsa and Ken Cope, Internet Programming (1995) and Marshall T. Rose, The Internet Message: Closing the Book with Electronic Mail (1993), which are incorporated herein by reference. Communications over the World Wide Web are discussed generally in John December and Neil Randall, The World Wide Web Unleashed (1996), which is incorporated herein by reference. Additionally, the illustrated embodiment is not limited to the specific networks known as the "Internet" and "World Wide Web", but relate to internet, intranet and web networks generally. A specific feature of this invention is that it is easily adaptable to control and automate communications via any type of communications network. In addition, it can select a preferred communications network and message encoding format to be used for a specific communications transaction, as further described below.

Detailed Description Text (7):
Appropriate programs executing on the provider computer 1 and the consumer computer 2 perform the functions necessary to transfer, maintain, and update the information at both locations. A program represents a set of stored instructions which are executed in a processor of the computer to process data, transmit and receive data, and produce displays. The provider program 12 operates to transmit changes in information stored in the provider database 11 at the provider computer 1. When changes are made to the information and the database, the provider program 12 operates to disseminate the changed information through the communications network 3. In the pushing method, the provider program 12 transmits the changed information, for example through e-mail, to the consumer computers 2 of all intended recipients. In the pulling method, the changed information is stored on a distribution server 32, such as a web server, which then can be accessed by the consumer computer 2. Any type of distribution server may be used, including network file servers, FTP servers, gopher servers, and so on. The type of distribution server used is not a limiting feature of the invention. The consumer program 22 will typically poll the distribution server 32 to determine whether the information has changed. This polling operation can be as simple as issuing a Web server HTTP file date request and comparing this with the file date of the last update. Polling is controlled by the information transferred from the provider program to the consumer program as further described below. Upon receipt of changed information, the consumer program 22 operates to perform certain functions with regard to that changed information. Principally, the information is stored in consumer database 21

on the consumer computer 2 for future reference and usage in controlling and automating communications between the consumer and provider. Furthermore, the information may be presented to a user at the consumer location, so that the user will be notified of the changed information. The information can be presented in a number of manners, including display or printing by the consumer program, sending an e-mail or voice-mail message to the user, paging the user, and other notification methods.

Detailed Description Text (8):
Since the provider knows what the changed information is and how consumers would likely prefer to be notified of the changed information, the transmitted information can include instructions on how the consumer program 22 should process the information for purposes of notification. For example, information from a provider may include the provider's telephone number. If the telephone number changes, the provider needs to supply everyone with whom it does business with the new number. The present invention provides a simple mechanism for carrying out such a data transfer, and of controlling which consumers receive overt notification. When the telephone number is changed in the distribution database at the provider computer 1, the information is transferred to the consumer computer 2, through either the push or pull method. Upon receipt, the consumer program 22 will process the changed information and store the new telephone number in the consumer database 21 for later access by the user or by other programs operating on the consumer's computer 2. At the consumer computer 2, the consumer may or may not be interested in overt notification of the new phone number; this depends on the consumer's relationship with the provider and how often and in what manner the consumer makes use of the phone number. This invention provides a way for notification to be cooperatively controlled by both the provider and consumer through the use of notification elements, which are described below.

Detailed Description Text (9):
Additionally, receipt and storage of the new or updated information can trigger other actions, such as automatically forwarding the information to another consumer, exchanging data with the consumer database 21, sending an automated response to the provider, or sending a message to another software program on the consumer's desktop. Again, this invention provides a means for such actions to be cooperatively controlled by both the provider and the consumer through the use of receipt methods, which are discussed below.

Detailed Description Text (10):
The information stored in the consumer database 21 can also include data, metadata, and instructions to be used by the consumer program 22 for controlling and automating communications between the provider and consumer. Again, because the provider of the information knows what communications response options are available to the consumers of the information, the provider can include the necessary data, metadata, and instructions to simplify and automate specific responses from the consumer to the provider. For example, the provider can include Web URL (Uniform Resource Locator) links to Web pages or forms on the provider's Web server. Or, the provider can also include special forms to be processed by the consumer program 22 that allow the consumer to automatically or semi-automatically transfer data from the consumer database 21 back to the provider. Examples include product order forms, survey forms, customer service request forms, scheduling forms, etc.

Detailed Description Text (11):
In the most general case, the provider knows what communications networks, network addresses, languages, encoding formats, data structures, and other communications processing data and methods are supported by the provider. Thus, the provider can include in the transferred information the data, metadata, and instructions necessary to control and coordinate general communications from the consumer to the provider or to parties related to the provider. For example, data, metadata and

instructions in the transferred information can be used by the consumer program 22 or other computer programs running on the consumer computer 2 to automatically format, compress, encrypt, address, and transmit copies of a word processing document, spreadsheet, database or database query, or other computer file format. Corresponding data, metadata, and instructions in the provider program 12 can control and automate the reception of the received message, including decryption, decompression, notification of the provider, and acknowledgment of receipt to the consumer. The same control technique can be applied to the execution of real-time communications, such as telephone calls, videoconferencing, or whiteboard applications.

Detailed Description Text (13):
Although any kind of data communications network and any kind of user interface can be used, the system can be constructed to work with existing Internet or World Wide Web protocols for data communications and display. In particular, the provider program and the consumer program can be designed to use HyperText Mark-up Language (HTML) for display and editing. HTML is discussed in Internet Request for Comment No. 1866, which is incorporated herein by reference. The use of HTML allows links to be made to other transmitted information or to other information accessible anywhere on the World Wide Web. Also, HTML forms can be used as an input mechanism. Standard Internet protocols for accessing the Web can also be used for accessing the information in the provider or consumer databases. To do this, the provider program and consumer program are designed to emulate a Web HyperText Transfer Protocol (HTTP) server. Then, any Web browser program conforming to the HTML/HTTP standard can generate Uniform Resource Locator (URL) requests to retrieve information from the provider and consumer programs and databases. A Web browser program is a set of instructions which causes the computer to execute information requests to various kinds of servers. The servers responded by transferring HTML files or other data files back to the browser program for display, processing, and storage. Protocols or formats other than HTML/HTTP can be used in the same manner, with an appropriate interface program for requesting, receiving, processing, and displaying data in accordance with the selected protocol or format. The operation of the provider and consumer programs in connection with the Web browser program is illustrated in FIG. 2. Since the provider and consumer programs operate identically in this regard, only the operation of the consumer program will be discussed.

Detailed Description Text (18):
Information can be stored in the provider and consumer databases 11, 21, transferred between the provider and consumer programs 12, 22, and processed by these programs in a variety of ways. The use of software objects and object-oriented databases, and in particular their ability to encapsulate data and methods for operating on that data in a single structure, provide certain degrees of functionality which are useful in the storage, transfer, and processing of information. For example, by using objects for transmission of the communications control files, and an object-oriented database for storage of these files, the received object can be stored by the consumer program 22 in its database 21 without having to disconnect and store the object's variables and methods independently. In addition, the data and methods of this object can be made available to other objects in the database or program for processing operations. Object oriented data structures, databases, programs, and processing are generally discussed in Grady Booch, Object Oriented Analysis and Design with Applications, (2nd ed. 1994) and James Rumbaugh, Object-Oriented Modeling and Design (1991), which are incorporated herein by reference. Thus, the following description of a preferred embodiment will discuss the use of objects. However, other methods for storing, transferring, and processing information, such as relational databases, binary files, or procedural programs, could be used.

Detailed Description Text (19):
As discussed above, the provider computer 1 includes a provider database 11 operated on by provider program 12, and the consumer computer 2 includes a consumer

database 21 operated on by consumer program 22. However, since "provider" and "consumer" are merely functional distinctions, in a preferred embodiment, a single computer and computer program would be able to operate as a provider computer 1 in executing instructions of the provider program 12 and as a consumer computer 2 in executing instructions of the consumer program 22. In this instance, only a single database may be used, if desired, to hold all of the data for transmitted objects and for received objects. The database structures described below could apply to a single database, or to separate databases if the programs operated separately. For ease of reference in describing operation of the provider program and the consumer program, separate databases will be illustrated.

Detailed Description Text (20):
FIG. 3 uses a standard object-oriented notational format to illustrate an embodiment of object classes in a single database 100 of the present invention. As shown in the global preferences object class 103, each object class includes three parts: an identifier 103A, an attribute section 103B, and a method section 103C. The method section 1 03C is used to perform operations on the attributes of the class. Class associations are shown with connecting lines. A plain line shows a one-to-one association. A line terminating in a solid dot shows a one-to-many association. A line terminating in a open dot shows a optional (zero or one) association. A diamond at the start of a line shows an aggregation association, i.e., the higher class contains the component classes. Inheritance between classes is shown with a branching line. Only certain attributes and methods are shown in object classes; many others have been omitted for clarity.

Detailed Description Text (27):
Type definitions provide a powerful tool for structuring the data included in a communications object, object update, or object message. This structured data provides the common "frame of reference" necessary to automate communications operations between a provider and consumer. It also allows communications objects to call each other's methods, and for other software programs to call the methods contained in communications objects stored in the consumer database 21. The latter technique requires the use of an Applications Programming Interface (API) which will be further discussed below.

Detailed Description Text (32):
Elements 143 may also be associated with methods 141 or rules 140. This is particularly useful for data exchange elements, which control the process of exchanging data between the consumer program 22 and the provider program 12 or another server program. Two specialized types of data exchange elements are useful for controlling the transmission of data external to the provider database 11. Attachment elements allow a provider to specify external files, objects, or other data stored in the provider's local or network computing environment to be included in the transmission of a communications object, communications object update, or message object. Query elements allow a provider to execute a query against a local or network database and have the results included in the transmission of a communications object. Data exchange elements, attachment elements, and query elements will be further explained in the description of data exchange control below.

Detailed Description Text (35):
Another example of a preference element is a notification element. Notification elements are used to control how a consumer is notified of new information when the object, object update, or object message is transferred. The format and structure of notification elements are discussed below in connection with the special processing notification elements receive. Any other consumer-editable preference regarding communications object attributes or method processing can be expressed as an preference element. Preference elements receive special processing by the consumer program 22 and storage in the consumer database 21 which will be further described below.

Detailed Description Text (36):
In addition to its composite type and composite value, each element 143 includes standard attributes such as system ID, name, description, version value, NewFlag, and HoldFlag. The system ID is a unique identification value in the database 100. Identification number assignments throughout the database are discussed below. The name is a label used to identify the element to the provider or consumer. The version value is used to coordinate updates each time the element is changed. The NewFlag is set to TRUE each time an element has been changed by a provider so that new information can be indentified for distribution by the provider program 12, and identified for updating when transferred to the consumer program 22. The HoldFlag is used to identify changed elements which are not yet to be distributed. The structure and content of elements may be more fully understood in connection with the description of notification elements discussed below.

Detailed Description Text (40):
The method class 141 is a form of metadata used to store methods which may be included in a communications object instance when it is transferred to a consumer. These methods should not be confused with the methods belonging to each of the other object classes in the system. A method object is primarily a mechanism for storing a method in the database for later inclusion in a communications object instance, at which time the method becomes a formal method of the communications object. Communications object methods are one of the most powerful aspects of communications objects. They allow the provider to specify processing instructions which will execute on the consumer's computer when certain conditions exist in the consumer program. For example, when a communications object is first received by the consumer program 22, a "receipt method" can automatically execute to return an acknowledgment message to the provider with information about that consumer transferred from the consumer database 21.

Detailed Description Text (42):
Instances of the method class 141 may implement communications object methods in several ways. The method can simply be a call to execute a system method included in the consumer program 22. The method can be actual instructions included in the object as program code in an executable format or an interpretable format, such as a script format. The method can be a call to the methods of another communications object located in the provider database 11 or consumer database 21. The method can also be a remote procedure call to another object or application located elsewhere on the consumer's computer or on a communications network 3 accessible from the consumer program. This remote procedure call can be executed at the remote computer, or it can be downloaded by the consumer program for local execution. The application of communications object methods to automating operations in the provider program 12 or consumer program 22 will be further discussed below.

Detailed Description Text (45):
Rules 140 work in conjunction with methods to provide the operational functionality of a communications object system. Rules allow the provider database 11 and consumer database 21 to operate as active object databases, capable of initiating communications, database processing, or other procedures based on time, system variables, system events, or other conditions. Rules also supply constraints under which methods operate. The usage of rules to control the operation of an active object database is discussed generally in Jennifer Widom and Stefano Ceri, Active Database Systems (1996), which is incorporated herein by reference.

Detailed Description Text (46):
Rules 140 consist of one or more conditions to be tested against. Rules 140 are associated with methods 141 to execute when the conditions are met. Rules 140 are typically expressed in boolean logic using standardized query languages or other appropriate formats. Rules can govern the behavior of individual communications objects, groups of communications objects (such as all objects from a particular

provider), all objects in the database, or general system actions. Examples of common processing actions governed by rules include backing up the database after X days or X number of changes, deleting or proposing for deletion communications objects that have not been accessed in X days, archiving X number of previous copies of a communications object or object component, using X amount of system resources when Y conditions are present, and allowing X number of communications objects or recipients under a particular software license. Rules may be understood further in the discussion of communications object control functions below.

Detailed Description Text (51):
The Recipient class 120 is used to determine the distribution of a communications object. Each communications object 110 is associated with one or more recipients 120 who receive an instance of the object when it is first created or when changes are made to it. Recipients are of two types: consumer programs 22, or distribution servers 32. A distribution server 32 may also be represented by a distribution service object. Distribution service objects are further discussed below. Transfer of communications objects 110 to both types of recipients is typically via the push technique. However recipients may also be tracked in the provider database 11 even if they use the pull technique of updating via the use of receipt acknowledgment messages. Acknowledgment messages are further described below. The push method may involve a fully automated transfer via a communications network 3, or it may involve a manual transfer such as a file copy over a network or via a computer floppy disk. Recipient objects 120 include the attributes necessary to generate and transmit an instance of the communications object to the recipient. To uniquely identify recipients even when names change, a SystemID attribute can used in addition to a Name attribute. System IDs are discussed below. Other attributes include the recipient's communications network address, such as an e-mail address, the type of encoding that should be used (e.g. MIME, BinHex, UUencoding, etc.), and the maximum attachment file size the recipient can accept (to determine if multiple attachments need to be sent). Recipients 120 have an association with methods 141 in order to allow different methods to be assigned to different recipients. An example is the communications object's update method. Communications objects transmitted to consumers via e-mail push may use one update method, while those transmitted to distribution servers may use a pull update method. Encoding methods, transmission methods, and other recipient-specific methods may also be assigned in this manner.

Detailed Description Text (54):
Four other classes in the database significantly involved with program operations are global preferences 103, report 105, folder 115, and event 116. Global preferences 103 provides a means for storing the preferences of a provider or consumer for general operation of the provider program 12 or consumer program 22. This may include attributes such as the default menu to display upon program startup, the default refresh interval to assign to new objects, the user's preference for notification when new objects arrive, the number of object archive copies the user wishes to keep, and other such preferences. Global preferences 103 may also include method preferences, such as the notification method to use when new objects are received, the method to use for archiving versions of objects or object components, and the method to use for backing up the database.

Detailed Description Text (55):
Report 105 is a class for storing report definitions and report display or printing preferences. As in many database management systems, reports may be defined by the system or by the user, and can include any listings, statistics, or analysis of value to the user.

Detailed Description Text (59):
In order to transfer a communications object instance or object update instance from a provider program 12 to a consumer program 22, the object must be output from the provider database 11 into a format suitable for transport via a communications

network 3. Any type of machine readable and writable format could be used, for example a compressed binary file such as that used by most relational or object-oriented database management programs. However, for maximum compatibility with communications networks 3 and other data processing systems, object instances can be written or read in an ASCII markup language, which is a superset of HTML. As with HTML, or other standard markup languages such as SGML, each item of structured data such as an object class or container class is expressed within a set of delimiters or "tags" defined in the markup language. Certain classes in the database structure exist specifically to provide the necessary container tags for other classes. For example, in FIG. 3, the methods 131, pages 132, elements 133, and type definitions 134 classes are all special container classes used to provide the tags necessary to delimit the methods, pages, elements, and type definition sections of an object output in the markup language. Another advantage of the use of an ASCII markup language is that the data and methods contained in communications object may be rendered readable to other data processing programs for purposes of interoperability. Other programs may also be programmed to output such a language or a subset thereof for purposes of importing into a communications object system program. The use of an ASCII markup language does not preclude the use of additional formatting or encoding, such as encryption, for the entire object or for portions of the object.

Detailed Description Text (62):
Because communications objects and their component type definitions, elements, pages, and methods are exchanged among multiple providers and consumers, the instances of these objects and components need to be uniquely distinguishable in each provider database 11 and consumer database 21. Name attributes alone cannot be relied upon to guarantee uniqueness. Other unique identification numbering systems could be employed, such as the provider's or consumer's U.S. Social Security numbers, U.S. Federal Employer Identification Numbers, passport numbers, etc. However, in a communications system which may be used globally, not all users may be assigned unique identifiers under one of these identification systems. A separate global identification system could be employed, such as the domain naming and e-mail addressing system used by the Internet. Although not all Internet users have their own Internet domain names, all of them have unique e-mail addresses. However, since users can and do change e-mail addresses, this would require that their system ID also change. The ideal communications system allows complete separation (or "abstraction" in object-oriented terminology) of a user's communications system ID from any real world names or physical communications network addresses with which the user is associated. In this way, users can change any of his/her names or physical communications network addresses while still maintaining complete continuity of his/her communications relationships. In addition, any changes to the user's name or physical communications network address can be automatically distributed by the user's communications object(s) to all other consumers with whom the user has a communications relationship.

Detailed Description Text (63):
To achieve this objective, a preferred embodiment of the present invention assigns a unique system ID value to each unique communications object and communications object component. This function is the equivalent of an automatically-generated unique key field ID in many conventional database management systems. This objective can be achieved in several ways. A first technique is to employ an algorithm that uses system state information together with data unique to the computer on which it is being run to produce system IDs whose probability of uniqueness is so high that for practical purposes they can be treated as unique. The use of such algorithms for creating globally unique object IDs is discussed generally in Kraig Brockschmidt, Inside OLE (1995), which is incorporated herein by reference. Standard industry algorithms and functions that have been created for this purpose include the Universal Unique Identifier (UUID) specified by the Open Software Foundation (OSF) Distributed Computing Environment (DCE) documentation. This algorithm is fully documented in Steven Miller, DEC/HP, Network Computing

Architecture, Remote Procedure Call RunTime Extentions Specification. Version OSF TX1.0.11 (1992), which is incorporated herein by reference. The use of a generally accepted industry UUID has the advantage of making a communications object identifier system directly compatible with other industry standard distributed object system specifications. Examples of such standards include the DCE and CORBA specifications from the Open Software Foundation and the OLE and DCOM specifications from Microsoft Corporation.

Detailed Description Text (64):
The second alternative is to use a centralized server or set of servers to produce unique system IDs as required. This approach has the advantage of creating a centralized registry of unique system IDs. Such a registry has other applications within a communications object system, as further described below. The architecture for centralized system ID assignment is illustrated in FIG. 5. A central system ID server 42 contain a database of system ID assignments 41. The system ID database 41 could be replicated across a group of ID servers 42 at various nodes of a communications network 3 to improve performance as the number of users increases. Upon initial installation, each provider program 12 or consumer program 22 sends a request 44 via the communications network 3 to the ID server 42 for a unique system ID 43. The ID server 42 returns a response 45 to the requesting program. The requesting program then saves the system ID in the provider database 11 or consumer database 21. This system ID 43 is shown in FIG. 3 as the SystemID attribute of the Database class 100. Within the database, the provider and consumer programs 12, 22 can include a function for assigning a separate unique system ID value to each instance of a communications object 110 or any class that will become a component of a communications object. In FIG. 3, these classes include the rule 140, method 141, page 142, element 143, and typeDefinition 144 classes. Again, this function is the equivalent of an automatically-generated unique key field ID in a conventional database management system. Since the provider's system ID 100 is unique for the entire communications system, and since each instance of a communications object system ID 110 or any or any component class system ID is unique within the provider's database, the combination of these system IDs creates a hierarchical indentification system capable of uniquely identifying every communications object instance or object component class instance throughout the communications system. This unique ID for any communications object or communications object component will be referred to as its UID.

Detailed Description Text (66):
The system ID assignment function can be modified to provide this capability by including nested system IDs for each group association within the system ID database 41. The object class model for nested system ID associations is shown in FIG. 6. The system ID database 250 contains any number of unique system IDs 251. Each of these may in turn contain zero, one, or more unique system IDs that function as group IDs as shown in association 255. This nesting of IDs may be as deep as necessary. Each system ID 251 includes a name and description attribute. For top level system IDs this would be the name and description of the provider. For lower-level group IDs this would be the name and description of the group (company, division, department, etc.).

Detailed Description Text (68):
The system ID server 40 shown in FIG. 5 is available system-wide, and includes at least one system ID object instance 43 in the system ID database 41 for each provider. Since this object instance contains the provider's name, description, and an authentication key, the system ID server 40 is a suitable mechanism to offer both system name directory services and system authentication services. These services are further described below.

Detailed Description Text (72):
In the case of a automatic HTTP request 37 from the consumer program 22 to the web server 32, the same MIME object transfer takes place, only the object is received

directly by the consumer program 22. In either case, the transfer to the consumer program 22 principally results in the execution of a set of processing steps. These steps typically include decoding of the MIME object, reading of the object, and storage of the object in the consumer database 21. The consumer program 22 can also execute other processing steps based upon the version of the object, the consumer's settings for preference elements in the object, other consumer preferences, and other methods in the object. The processes for storing and processing communications objects are discussed below.

Detailed Description Text (73):
FIG. 8 illustrates transfer of an object through e-mail using the push technique. The browser program 50 is not used for this function. The object may be attached as a MIME object to an e-mail message 38. Other attachment or encoding types may be used, such as BinHex or UUencoding. The object may also be encoded in ASCII within the text of the e-mail message itself. The optimal encoding method for each recipient can be selected and employed automatically by the provider program 12 when this information is included in the Recipients (120, FIG. 3) class, as further described below. The transmission steps for each attachment or encoding type may vary slightly. The transmission steps for a MIME attachment will be described here. The e-mail message is sent in the ordinary manner, using whichever e-mail servers and intermediaries are available (i.e., through the Internet 3), to reach the consumer's e-mail server 31. The consumer's e-mail program 62 retrieves the mail message from its server in the ordinary manner. Depending upon operation of the e-mail program, the attachment may be downloaded for storage in either an internal or external MIME directory 63, 64, or left for storage on the e-mail server 31. The consumer program 22 then periodically polls the MIME directory 65, 66 or the e-mail server 31 to locate objects of a communications object MIME type. If a communications object type is located, it is read from the storage location and processed by the consumer program as described below. It may also be deleted from the MIME storage area by the consumer program 22 after it has been read and processed.

Detailed Description Text (77):
Broadcast networks such as television or cable systems can represent particularly efficient means of transmitting communications objects or object updates via the push technique if the consumer computer 2 is equipped with a device for receiving and decoding the broadcast signal. By applying the filtering techniques described above to "listening" on a broadcast network, a consumer program 22 can receive only the communications object updates intended for communications objects in the consumer's database 21. Because broadcast networks are transmit-only, communications back to the provider must be accomplished using a "back channel" such as a telephone network or computer network, e.g. the Internet.

Detailed Description Text (79):
As described above, the provider program 12 operates as a state machine in generating HTML screens and forms which are displayed by the user's browser program. The provider program 12 is used to create and edit instances in the provider database 11 of the object classes described above. The provider program 12 is also used to publish and distribute instances of communications objects to one or more consumer programs 22 or distribution servers 32 through the communications network 3.

Detailed Description Text (81):
The first five choices on the main menu 300 allow the user to work with the communications objects, pages, elements, type definitions, methods, and rules stored in the provider database. The provider program is primarily creating, displaying, editing, and reporting on objects in the provider database. Therefore, the menus and forms used by the provider program are similar to a the menuing, browsing, editing, or reporting modes of any conventional database application. Initially, there are no user-defined communications objects, pages, elements, type

definitions, methods, or rules. (System-defined communications objects, pages, elements, type definitions, methods, or rules may exist but are not editable by the user). Upon selection of one of the menu choices, a HTTP request is generated to display the requested HTML page. The communications object 320, page 330, element 340, type definition 350, and method/rule 360 forms include similar functions: create, edit, delete, and preview. Although the functions are similar, each menu has links to different HTML forms used for performing the functions on the different types of data (communications object 321-324, page 331-334, element 341-344, type definition 351-354, and method/rule 361-364). In addition to the menu choices, a list of the appropriate class instances from the provider database 11 is displayed in order to select the data to edit, delete, or preview. In one embodiment, hyperlinks or form buttons for editing, deleting, and previewing are associated with each data item in the list. Alternatively, a single link to the edit, delete, or preview forms can be used and the data item can be selected from a list when the appropriate form is displayed.

Detailed Description Text (83):
FIG. 10A shows the processing steps to be taken upon submission of a create form. These steps also apply to submission of an editing form as described below. When a create form is submitted (step 400), the provider program 12 first determines whether the form data is valid (step 401). If it is not the provider program returns an error screen or form with information about the error to the user (step 411). This error screen may include a form for correcting the error, or hyperlinks to other forms where the error can be corrected. Once a form passes the validation test, the provider program then determines whether the form is a create or an edit operation (step 402). For a create operation, the program next assigns the new instance an initial version value (step 403), sets the instance's NewFlag attribute to TRUE (step 404), and saves the instance to the provider database 11 (step 405). The version value is used to compare changed object class instances in the object reception processing. The NewFlag attribute is used to indicate a class instance that requires distribution.

Detailed Description Text (84):
The submission of any new or changed data in an communications object component class triggers the update association rule. This rule can be stored as a rule 140. The method associated with this rule is illustrated in FIG. 1 OB. In this method, the provider program first queries the provider database 11 for all associated class instances which contain the new or updated class instance (step 431). The program then processes each associated class instance to determine whether it is already identified as a new instance (steps 432, 433). If the associated class instance is not new, the version value is incremented (step 443), the NewFlag is set to TRUE (step 442), and the instance is stored in the provider database 11 (step 441). When an associated class instance becomes new, every container association with this instance must also be processed (steps 431-433). In this way the program processes the entire tree of all class instances which contain the newly created class instance, incrementing version values and marking them as new. This is necessary to ensure the complete distribution of all associations to any new or changed class instances. When all container class associations have been updated, the next HTML screen is generated (step 435).

Detailed Description Text (85):
The edit forms 322, 332, 342, 352, 362 shown in FIG. 9 permit the editing of instance attributes and associations in the database of the appropriate class. For example, the communications object edit form 312 will list the pages which currently exist in the database and therefore can be assigned to the object. A submitted edit form 322, 332, 342, 352, 362 is processed according to the steps illustrated in FIGS. 10A and 10B. A test for an edit form is performed in step 402. From this point there are only two differences from create form processing. First, if the NewFlag attribute of the edited class instance is already TRUE (step 412), this means the instance has been edited since the last distribution operation. In

this case, the update association method need not be performed. The edited instance can simply be saved to the database (step 425) and the next HTML screen generated (step 426).

Detailed Description Text (86):
Second, edited instances do not necessarily replace the previous instance when stored in the database (steps 415, 425). Multiple versions of object instances may be maintained in the database so that the user can revert to previous data. The number of previous versions stored is controlled by a global preference attribute (103, FIG. 3) or a communications object preferences attribute (127, FIG. 3) and one or more archiving rules 140. Each time an edit form is submitted, the archiving rule 140 is triggered. Using the appropriate preference attribute, the archive rule determines if the preferred number of previous versions of the communications object (110, FIG. 3) are already stored in the database. If so, then the oldest version is deleted when a new version is stored. If not, the new version is added to any previous versions that exist. Data archiving control is further discussed below.

Detailed Description Text (87):
The delete forms 323, 333, 343, 353, 363 shown in FIG. 9 are used to remove class instances from the database. The form can require confirmation that the selected instance is to be deleted. Additionally, the delete form can provide a list of other instances of the same class in order to allow the selection of multiple items for deletion. Processing of a submitted delete form first involves executing the steps of the update association method illustrated in FIG. 10B. Then the selected class instance or instances can be deleted from the database. Instance deletion may follow a user preference for archiving a specified number of deleted instances, or maintaining deleted instances for a specified interval of time before purging them completely from the database.

Detailed Description Text (89):
The recipient form 310 accessed from the object menu 320 is used to assign the recipients (120, FIG. 3) who will receive each communications object. From this form the user can add or delete recipients associations for the selected object by the use of checkboxes for each recipient. The user can also choose to go to four additional forms. The create recipient form 311 allows the user to add a new recipient instance to the database. The edit selected recipient form 312 allows the user to edit the recipient's settings for communications object distribution. The delete recipient form 313 permits the user to delete a recipient from the database. No special processing is required when adding, editing, or deleting recipient instances since they are not a communications object component and there are no associations that contain them. However, NewFlag and HoldFlag attributes for recipients are set as described previously for purposes of communications object distribution. The preview recipient form 314 allows the user to see precisely how any selected communications object and its component pages and elements will appear to a selected recipient.

Detailed Description Text (90):
The reports form 370 is used to create, edit, delete, and display reports (120, FIG. 3) from the database. Menu items link it to the create report form 371, edit report form 372, delete report form 373, and display report form 374. The preferences form 316 is used to edit the user's overall preferences (GlobalPrefs 103, FIG. 3) for program display and operation.

Detailed Description Text (94):
FIG. 11 illustrates the process performed by the provider program 12 in distributing an entire object. The provider database 11 is queried (step 501) to determine all new or changed communications objects which need to be published, i.e., those which have a NewFlag attribute set to TRUE and a HoldFlag attribute set to FALSE. The program then loops (step 502) through each communications object

instance 100 which is to be published. For each object the program reads the associated recipients 120 (step 503). The program begins a second loop (step 504) through each recipient 120. Using the recipient attributes and methods, a communications object instance is generated and transmitted to this recipient (step 505, further shown in FIG. 12). The loop is repeated for all recipients of the communications object.

Detailed Description Text (95):
At this point all new or changed communications objects have been distributed to their assigned recipients. However, a new or edited recipient may need to receive an existing communications object that has not changed. To account for this case the provider program queries the database for all recipients 120 whose NewFlag attribute is TRUE and HoldFlag attribute is FALSE (step 511). The program then loops (step 512) through each of these recipients 120, quering for the associated communications objects 110 where the NewFlag attribute is FALSE (step 513). The program then loops (step 514) through each of these communications objects 110 executing the object generation and transmission routine for each object (step 515).

Detailed Description Text (96):
After all communications object instances 110 have been transmitted, the program does another query of the database for all class instances where the NewFlag attribute is TRUE and HoldFlag is FALSE (step 521). The program loops through these instances and resets their NewFlag attribute reset to FALSE (steps 522, 523). This prepares the database for the next round of editing and publishing.

Detailed Description Text (97):
The procedures for generating and transmitting the communications object instance for each recipient are illustrated in the flow chart of FIG. 12. The program begins by creating (Step 531) a header portion of an object markup file from the attributes of the communications object (110, FIG. 3). A header portion includes a header tag, the provider's system ID (the SystemID attribute of database 100, FIG. 3), and any group IDs or category IDs (250, 251, 252, FIG. 6), the communications object 110 system ID attribute, and other attributes of the communications object appropriate for transmission. (Note that the combination of the provider's system ID and the communications object 110 system ID consitutes the communications object's UID described previously.) Next, the program reads all type definitions (144, FIG. 3) associated with the object (step 532) and writes them in the markup language format to the markup file (step 533).

Detailed Description Text (102):
After encoding, the communications object is transmitted to the recipient according to the attributes and methods of the recipient (steps 550-551). As discussed previously, according to a preferred embodiment, objects sent directly to consumer computers 2 using the push method are sent as e-mail messages or message attachments to the addresses of the recipients. These transmissions can be queued using scheduled events 117 to reduce system load. Objects sent to a distribution server 32 for distribution using a pull method are saved to the appropriate web server document directory. Alternatively, based upon the access the provider has to the provider's web server, the object could be mailed to the Web server administrator, uploaded as an HTTP form to the Web server, or otherwise stored for later posting by the Web server administrator. The transmission steps could also include an e-mail message, voicemail message, or other notification to the administrator that the object is ready to be stored on the server. Alternatively, transmission to a distribution server 32 can be automated through the use of a distribution service object. Distribution service objects will be further discussed below.

Detailed Description Text (105):
One advantage of the communications system of the present invention is that the

transmitted communications object instance can be automatically received, processed, stored, and indexed by the consumer program 22. Since the data is structured as an object and stored in an object-oriented database 21, the data it contains can be easily searched using the consumer program 22 in order to locate specific information or perform certain functions.

Detailed Description Text (107):
FIG. 13 illustrates the relationships between various screens and forms produced and used by the consumer program in processing objects stored in the consumer database. The consumer program is primarily reading, editing, and reporting on objects to the consumer database. Therefore, the menus and forms used by the consumer program are similar to a the browsing, editing, or reporting modes of any conventional database application. Upon startup, an HTML page of the main menu 600 screen is generated and displayed. As with the provider program 12, the menus and forms discussed with respect to the consumer program 22 are merely illustrative of the capabilities of the system. They can be organized in any order or hierarchy, and other functions and features can be added by creating or modifying other menus, forms, or toolbars.

Detailed Description Text (108):
The main menu 600 lists the principal types of functions which can be performed by the consumer program 22. The object list form 610 provides a directory to the communications objects in the consumer database. A name or other identifying information for each object is displayed in a list format. The name or identifying information also functions as a hyperlink to the object. The user can set various attributes of the display, such as formatting of characters, amount and order of information identifying the object, and organization of the communications objects in the list, using the preferences form 650. The choices in the object list form provide access to forms for performing functions with respect to the attributes or methods of one or more communications objects selected in the object list.

Detailed Description Text (109):
The search form 620, as illustrated in FIG. 14 will be used as an example for processing of a form request. The search form 620 presents the user with a screen which allows the input of information, whether typed in or selected as check boxes. As illustrated in FIG. 14 the search form 620 includes a location to enter a search string. The search may also be expanded or limited based upon the form from which the search form was requested. For example, if the search form 220 was selected from the selected object form 211, then only that selected object is searched. The user may elect to search all objects instead by checking an appropriate checkbox. Similarly, the search can be limited to certain folders of communications objects. The user can also select the method for display of the search results. When the search form 220 is submitted as a request, the consumer program 22 will then act to process the form (step 57 in FIG. 2). The processing of a search form results in a query of the consumer database 21 according to the search attributes entered in the form. The query is performed in the same manner as queries for any object-oriented or relational database. A search report is then generated as the next screen (step 53 in FIG. 2), which is outputted to the browser program 50 and displayed (step 54 in FIG. 2). In the search report, the consumer program 22 will automatically generate a hyperlink URL for each communications object name and page name displayed so that the respective object and page can be selected.

Detailed Description Text (111):
The sort form 634 presents a set of options for displaying communications objects, pages, and elements. Choices include sorting by container (such as a folder), order (ascending or descending), and unit (object, page, element). The class instances in the consumer database 21 are then sorted according to the selected criteria and redisplayed.

Detailed Description Text (112):

The export form 645 operates to transfer data from the database to be used by other applications, such as a contact file for a personal information manager or a mail merge list for a word processor. First, a search or sort is performed to select a group of communications objects, pages, or elements to be exported. The export form includes choices to select the elements to export, the destination (such as a disk, file, clipboard, etc.) and a format. Upon submission of the completed form, the data meeting the export form criteria is transferred to the selected destination in the selected format. The data can then be used by the other application. A screen identifying the results of the export is then displayed.

Detailed Description Text (113):
The print form 646 is used to print information in the database. Some routine print functions can be performed by the browser program 50. However, other printing functions, such as printing selected elements or pages or using special print formatting, can be performed directly from the print form. The print form requests information relating to the selection of elements to be printed and the format for printing. A results screen can also be displayed after the print operation.

Detailed Description Text (115):
An edit object form 622 can be used to edit a communications object's attributes, including its component elements. Most attributes and elements of a communications object are defined by the provider and are not editable by the consumer. However, certain elements are defined by the provider specifically for editing by the consumer. These preference elements may include polling refresh intervals, return receipts, subscription elements, and notification elements. A consumer may also assign other attributes and associations to a communications object. These include folder assignments, nicknames, notes, notification priority, expiration date, and archive method. All communications object attributes and element attributes edited by the consumer are stored separately from the object in the consumer database 21. This is accomplished by use of the CommObjectPrefs class 127 and ElementPrefs class 147 shown in FIG. 3. Whenever the consumer first edits or adds communications object attributes, an instance of the communications object preferences class 127 is created in the consumer database 21 and associated with the communications object 110. Similarly, whenever the consumer first edits a preference element, an instance of the element preferences class 147 is created and associated with the element 143. The edited or assigned attributes are stored in these two class instances, and appropriate methods 141 are stored with or associated with these classes (these associations are not shown in FIG. 3 due to space limitations). In this way the consumer's data is not overwritten when an updated communications object is received. Additionally, the consumer may forward a communications object without including the consumer's own attribute preferences, although the consumer may optionally choose to do so. Communications object forwarding is described further below.

Detailed Description Text (116):
The delete object form 623 shown in FIG. 13 allows a communications object to be removed from the consumer database 21 if the information is no longer desired. The form also allows the consumer to reconfirm that the selected object is to be deleted. Additionally, the user may select certain options for deletion. Such options may include maintaining the object for a predefined period before actual deletion, or storing basic information (such as an object name, UID, and update method) so that the object could easily be retrieved again if needed.

Detailed Description Text (118):
The notification report form 630 is selected from the main menu 600 in order to provide information about new communications objects, updates to existing objects, messages received by objects, database status messages, and other news of which the user wishes to be informed. The notification report form provides the user with the capability to select and filter information received from a provider. Operation of this form is discussed below in connection with notification element processing.

Detailed Description Text (119):
The user can generate other reports relating to the consumer database using the other reports form 640. Standard reports might include database statistics (total objects, pages and elements; database file size; and size of objects being held), object statistics (frequency of use; last use; age in system; total age; size; number of updates; and last update), and transaction logs (number of updates; percentage of CPU time used, online time used; percentage of errors; and types of errors). Additionally, consumers could specify their own database reports to be added to this form.

Detailed Description Text (120):
The preferences form 650 allows the user to edit operational preferences that apply to the consumer database 21 or consumer program 22 as a whole. These can include configuration options such as the initial menu to display upon startup, the colors and fonts for the forms and data, and field defaults. The user may also select options such as a default refresh interval to use for new objects, a default expiration period, and default settings for editing or preference forms.

Detailed Description Text (122):
FIG. 15 is a flow chart illustrating the operations for processing communications object instances 110 received by the consumer program 22. As shown, an entire object is provided (Step 700) to the consumer program 22 each time any changes are made to the object. Alternatively, only the changed portions of the updated object may be sent in an object update. These processing steps for this case are not described but are substantially similar. Upon receipt of the object, the consumer program 22 first determines whether the received object is a message object (step 701). Message objects and their processing will be explained below. If the received object is not a message object, the next step is determining whether the communications object already exists in the consumer database 21. This is done by querying (step 702) the consumer database 21 for the UID of thfe communications object 110. If the UID does not exist, the object is processed as a new object.

Detailed Description Text (123):
For new objects, the consumer program 22 first executes the consumer's GlobalPrefs NewObjectReceipt method (step 703). This method allows the consumer to control the processing of new communications objects. Typically this method will store the object to the consumer database 21. However, the consumer may wish to discard objects received from any provider system ID on a list maintained in the consumer database 21, commonly referred to as a "kill file". Additionally, the NewObjectReceipt method controls the permissions the consumer extends to the new object to execute its own receipt method(s). For example, new objects from providers whose system ID is not in the consumer database 21 may not be allowed to execute their receipt method, while new objects from known providers may be extended this privilege. Receipt methods trigger automatic actions taken when a communications object or object update is first received by a consumer program 22. For example, a receipt method may automatically return an acknowledgment message back to the provider confirming the consumer's receipt of the object or object update. Receipt methods and acknowledgment messages are further described below.

Detailed Description Text (124):
Once the NewObjectReceipt method has been executed, the consumers notification preferences for new objects are retrieved (step 704) from the NewObjectNotify attribute of the GlobalPrefs class (103, FIG. 3) in the consumer database 21. A test is done to see if notification is desired (step 705). If so the consumer program 22 retrieves and executes the consumer's GlobalPrefs NewObjectNotify method (step 706). The user may wish to have the object displayed immediately, to receive an e-mail about the new object, to include a message about the new object in the user's notification report (including its size, methods, update intervals, etc.), or any other notification action or combination of actions. Notification

preferences and methods are further described below. Also, different actions may be taken based upon the program state and operation involved with the object's arrival. For example, the user may wish to have an object displayed immediately if the user manually selected it as a HTTP request from a Web site, but not if it was an object update retrieved automatically via a Web HTTP polling request by the consumer program 22, or if it arrived via e-mail. Different actions may also be taken based upon attributes or methods of the communications object itself, or a comparison between these and with the existing objects in the consumer database 21. For instance, the consumer may wish to immediately display new objects from selected providers whose system ID is already present, but only have notification in the notification report of new objects from any other provider system ID.

Detailed Description Text (125):
After any notification methods have been executed, the consumer program 22 executes any other system methods that may apply to new communications objects (step 708). For example, a Register method would check to see if the updated object wished to register a new public method (141, FIG. 3) in the consumer database 21. After any system methods are executed, new communications object processing is complete.

Detailed Description Text (126):
In step 702, if an object already exists in the database, then it is processed to determine what changes have occurred and what actions should be taken by the consumer program 22 because of those changes. In this way the communications control system of the present invention functions not just as an information transfer system but as an event processing system. Both the provider and consumer share control over the processing that takes place when knowledge of an event is transferred from provider to consumer. The first event, the arrival of a communications object update, is processed in step 714. The version value of the updated communications object 110 is compared with the version value of the most recent version stored in the consumer database 21. In the authoring process, the update association rule and method (FIG. 10B) has ensured that any change to a component of a communications object results in the communications object's version value being incremented. Therefore if the newly received object's version value is older or equal to that of the existing object, the newly received object is not new. In this case the object is either discarded, or other processing may take place depending on the consumer's preferences, such as notification in the consumer's notification report (step 713). Communications objects with equal or lesser version values typically represent retransmissions due to distribution errors by the provider, forwarded objects from other consumers, or manual retrievals of an object by the consumer when the consumer is unsure of the object's update status.

Detailed Description Text (127):
If the newly received communications object's version value is newer than the last version stored, the consumer program 22 first stores the updated object in the consumer database 21 (step 715). The next set of steps involves updating communications object associations. When a consumer is able to edit data related to a communications object, this data needs to be stored separately from the communications object so it is not overwritten by a subsequent communications object update. The data structures necessary to accomplish this are shown in FIG. 3. First, when a consumer first edits any attribute relating to a communications object 110 as a whole, an instance of the communications object preference class 127 is created by the consumer program 22. This instance 127 has a one-to-one association with its parent communications object 110. It also has a one-to-many association with folder instances 115. These associations are created using the edit object form (622, FIG. 13), and allow the consumer to further control processing related to this communications object. A consumer is also able to edit preferences related to specific elements 143 within the communications object 110. As described above, these preference elements are a mechanism for providers to give consumers control of specific types of communications object update processing.

Whenever a consumer edits a editable preference element 143, an instance of the element preference class 147 is created by the consumer program 22. This element preference instance 147 has a one-to-one association with its parent element 143. Optionally it may also have an association with one or more folder instances 115, which allow the consumer to further control processing related to this preference element. When an updated communications object is received by the consumer program 22, the associations between the communications object preference instance 127 and each element preference instance 147 need to be updated to the new "parent" communications object 110 and elements 143. These update steps are shown as steps 716 and 717 of FIG. 15. Referring again to FIG. 3, if an element 143 for which an association with an element preference 147 exists is absent in the communications object update, the consumer may wish to be notified and/or the element preference instance 147 deleted. This can be accomplished via a notification method as described below.

Detailed Description Text (129):
After receipt method processing, notification processing is carried out. Processing of notification elements (steps 723-727) is further described below. After notification processing, the consumer program 22 executes any other system methods that apply to updated communications objects, such as the Register method (step 731). Finally, the consumer program 22 checks the archive preference attribute of the communications object preference instance (127, FIG. 3) to see if it exists (step 735). Archive preferences determine the number of previous instances of a communications object stored in the consumer database 21. This is identical to how archiving works for previous versions of communications object components stored in the provider database 11. In a preferred embodiment, consumers can control archiving either globally or by individual communications object. If the consumer has indicated an archive preference for the object, the consumer program 22 executes the archive method indicated by the communications object preference (step 736). If no such archive preference exists, the consumer program 22 executes the archive method indicated by the consumers global preferences (103, FIG. 3) in step 737. This completes the processing of an updated communications object.

Detailed Description Text (131):
The functions of provider and consumer programs and databases have been separated in the above discussion in order to simplify the description of the communications system of the present invention. However, in one embodiment, the program functions and databases are combined. Thus, a single database includes all of the communications objects and object components which have been created or received by the user. This eliminates complexity and saves disk space for the user. The program offers the provider functions when creating and distributing communications objects, and the consumer functions when receiving and processing them. Combining the program functions and databases in this way yields significant additional functionality not available when the programs and databases are separate.

Detailed Description Text (133):
Second, all the elements, type definitions, and methods of both received and transmitted objects are present in a single database and program operation environment. This allows the provider to use the attributes and methods of received objects for other purposes. For example, special communications object types can be used to supply services needed by other communications objects. Such services include directory services, authentication services, payment services, and feedback services. These "service objects" will be further discussed below. Components from received communications objects can also be reused within the provider's own communications objects, thus creating "synthesized objects". Synthesized objects will be further discussed below.

Detailed Description Text (145):
Communications objects represent a transfer of communications intelligence, in the form of data, metadata, and instructions, from a provider to a consumer who wishes

to form a communications relationship with that provider. Once the communications object has been exchanged, further communications between the provider and consumer can carry greater intelligence because they can be be orginated and received as transmissions between these two communications objects. Although these messages can be structured in any form, in a preferred embodiment they are simply a special communications object type called a message object 110. This means they can be generated, encoded, transmitted, received, and processed in the same fashion as any other communications object. The only difference is that the generation or receipt of a message object may not result in an update to the sending or receiving communications object, but rather the execution of one or more methods at the sending or receiving program, and optionally changes to other objects or object components in the sending or receiving databases. A communications object update may be considered a special form of message object which includes changes to the receiving communications object.

Detailed Description Text (148):
The processing steps for receipt of a message object by the consumer program 22 are illustrated in FIG. 15. First, a newly received object is tested to determine if it belongs to the message object subclass (step 701). If so, the next test is to see if the message object's "parent" exists in the consumer database 21 (step 711) by searching for its UID. The parent is the communications object (110, FIG. 3) that produced the message object. If the UID of the parent object is not present, the message object is rejected as invalid. This may also result in an error message being displayed to the user or placed in the the user's notification report, depending on the user's preferences.

Detailed Description Text (149):
If the parent UID exists, the final step is to execute the message object's receipt method or methods. Since a message object is simply a special type of communications object, it may carry its own methods, or it may call the methods of its communications object "parent". Additionally, when received by the originating provider program 12, it may call any other method 141 present in the provider database 11 which originated the parent communications object. For example, a communications object could include a receipt method 141 named GetStatData which obtains statistical data from a consumer database 21 and returns a message object to the provider program 12. When the message object is received by the provider program 12, it may execute a receipt method 141 named PostStatData which is present in the provider database 11, but not in the original communications object 110. Alternatively, method names can be polymorphic. In this case a method included in the communications object 110 could perform one action when received by the consumer program 22, but another action when called by a message object in the provider program 12. The method can distinguish between these programs by matching the system ID of its originating database (100, FIG. 3) with the system ID of the database in which it is executing. (Such a match may also be made on a group ID rather than a specific system Ib.) For example, a communications object 110 could include a method TransferStatData which, when executed by the consumer program 22, would be used to gather statistical data from the consumer database 21 and return a message object to the provider. However, when the same TransferStatData method is executed by the message object back at the provider program 12, the method could be used to post the statistical data to the provider database 11 (or another database maintained by the provider).

Detailed Description Text (150):
Because of this, message objects generally do not need to transport their own methods, but can instead call methods present in their parent communications object (already stored in the consumer database 21 and provider database 1 1), or other methods from their originating provider database 11. This makes them a highly efficient means of transporting structured message data and initiating automated processing of that data.

Detailed Description Text (154):
Each component object has an association 110A to the composite object which contains it. A component object may be contained by more than one composite object. As with other associations in the provider database 11, changes to component objects can be propagated upward to the composite objects which contain them via the update association method (FIG. 10B). Thus, for editing and display purposes, component objects can be treated as one or more methods, rules, pages, elements, or type definitions that become part of the larger composite communications object. Composite objects can access the elements or methods of their component objects in the consumer database 21 just as if the elements or methods were contained directly in the composite object. In this manner component objects can be dealt with as independently transferrable objects for purposes of updating, distribution control, and other uses as described below, while still functioning as an integral part of the composite objects which contain them.

Detailed Description Text (156):
Composite and component objects are particularly useful for creating many different kinds of metadata structures in communications object system databases 100. Example are directory category hierarchies and discussion response thread hierarchies, shown in FIGS. 29A and 29B. Another example is schedule objects

Detailed Description Text (158):
When the functions of the provider program 12 and consumer program 22 and their respective databases 11 and 21 are combined, a provider can create synthesized objects. A synthesized object is a communications object which contains components or component objects from other providers. These are referred to as "external components" or "external component objects". FIG. 17 illustrates the synthesized object subclass 813. A synthesized object does not necessarily require a special communications object type, although it may be desirable for licensing, authentication, or other purposes. The uses of synthesized communications objects may be more fully understood from the description of service object types below.

Detailed Description Text (161):
Service objects are another special class of communications object whose primary function is to provide communications services to other communications objects. As shown in FIG. 17, the service object superclass 815 can be further broken into subtypes such as registration service objects 830, maintenance service objects 831, name service objects 832, directory service objects 833, and so on. Service object types can be used by the provider program 12 and consumer program 22 to distinguish the services that service objects make publicly available to other objects. The use of service objects will be discussed in separate group of sections below.

Detailed Description Text (163):
User objects are communications objects 110 used to represent communications object system users or groups of users in a communications object system database 100. User objects are shown as class 816 of FIG. 17. User objects 110 have many different applications for both service object partner servers and multiuser implementations of a communications object system database 100. User objects will be further discussed in the service object and advanced system architecture sections below.

Detailed Description Text (165):
Schedule objects are communications objects 110 used to represent scheduled real-world events in a communications object system database 100. Scheduled objects are shown as class 817 of FIG. 17. Schedule objects 110 are used to coordinate communications about events, such as phone calls and meetings. Schedule objects will be further discussed in the scheduling control section below.

Detailed Description Text (175):
The second case covers how the provider can allow the consumer to control

distribution using the push technique. For example, a software company might offer multiple pages within a communications object pertaining to a software product. Each page would correspond to a particular version of that product. If the company did not know which version a consumer was using, it could include a menu of these pages in the communications object. A consumer's choices from this menu would be automatically returned to the provider via a message object, which would invoke a method in the provider program 12 to change the consumer's page subscription settings in the provider database 11. In this way the consumer can choose to "subscribe" to the page or pages corresponding to the product version the consumer is using. This saves transmission time for the provider and file space for the consumer.

Detailed Description Text (177):
Once the communications object is received by the consumer program 22, the SubscribeFlag values of the page subscription elements 853 are editable by the consumer using the edit object form (622, FIG. 13) (The operation of this form in conjunction with the operation of the consumer program 22 is described above.) When this form is submitted to the consumer program 22, its contents are processed by the associated PageSubscribe method. This method first creates a message object instance (810, FIG. 17) containing the changed page subscription elements 853 and the receipt method PageSubscribe. It then transmits this message object instance 810 back to the provider program 12. When the message object instance 810 is received by the provider program 12, the PageSubscribe receipt method is executed using the changed page subscription elements 853 as a parameter. As a polymorphic method, this results in an update operation that changes the SubscribeFlag values of the page subscription elements 853 associated with the recipient 120. This in turn removes the association 856. In this way the consumer is able to edit his/her own page subscription settings in the provider's database 11. An updated communications object instance can be returned by the provider program 12 immediately, at a scheduled future date/time, or at the time of the provider's next publishing operation.

Detailed Description Text (181):
The next step is for the consumer to obtain a copy of the composite communications object instance 900 (step 930). When the object is received by the consumer program 22, the distribution control method 931 can be executed automatically as a receipt method or manually by the consumer (step 931). The distribution control method 931 determines which component objects 901 should be retrieved. These instructions may incorporate any logic or business rules the provider wishes to employ, using whatever data is available to the communications object in the consumer database 21 or elsewhere in the consumer's computing environment. For example, if the communications object represented a software product, the distribution control method 931 could examine the consumer database 21 or the consumer's local or network environment to determine if the product was installed and what version the consumer was using. Alternatively it could present an input form to the consumer to gather other relevant data for processing. The distribution control method 931 could then determine and download the appropriate component objects 901 from the distribution server 32 (step 932). For example, it could download the component objects 901 that correspond to the version of the product the consumer was using. If the consumer did not have the product installed, the distribution control method 931 could download the component objects 901 that are compatible with the consumer's computer system. Alternately, the distribution control method 931 could transmit data it retrieves from the consumer database 21 or the consumer's computer environment to the distribution control object 902 on the distribution server 32 (step 933). This data could then be processed by the distribution server 32 to determine the optimal component objects to return to the consumer program 22. This can be more efficient than transferring a sizable distribution control method to each consumer. Automatic data exchange will be further discussed below.

Detailed Description Text (186):

Encoding refers to the formatting of communications data to increase its communications value. Communications encoding may take many forms, including human languages, computer languages, character sets, data file formats, compression formats, transmission formats, encryption formats, and display formats. Multiple types of encoding may be applied to the same communications transmission. A communications object system represents a significant improvement over existing communications encoding control processes for three reasons. First, communications objects provide a simple, automated way for the communications sender to know which encoding formats are optimal for a communications recipient. Secondly, because this encoding data is stored within a structured database 11, 21, it can be easily accessed by the provider program 12, consumer program 22, or another software program using an appropriate API, for the purposes of automating both the encoding process for the sender and the decoding process for the receiver. Thirdly, the sharing of encoding and decoding methods can be dramatically simplified through the use of encoding service objects. Encoding service objects will be further discussed below.

Detailed Description Text (191):
As with other forms of encoding, communications objects are an excellent mechanism for simplifying and automating public/private key encryption. Referring to the data structures in FIG. 3, this is because a communications object 110 is an ideal vehicle for transmitting one or more of the provider's public keys to the consumer's computer, where it can be used to automatically encrypt messages being returned to the provider. The public key can be stored as an element 143, and the encryption method can be stored as a method 141. By encrypting the return message as a message object 1 10, the message object can invoke a receipt method 141 at the provider program 12 which can automatically decrypt the message using the provider's private key and the decryption method, stored as an element 143 and method 141 in the provider database 11, or otherwise made available to the receipt method 141.

Detailed Description Text (197):
When the e-mail message 956 is received by the provider program 12, the WPFileSend method stored in the provider database 11 is executed. This performs each decoding step in reverse order. First the e-mail message 956 is decoded to produce the message object attachment or attachments (step 971). Then the message object or objects are read and processed to determine the subsequent decoding steps necessary (step 972). Using the function calls and parameters supplied in the message object, the word processing file or files 952 are decrypted (step 973). The same procedure is followed for decompression (step 974). Next, the file or files files are saved to an appropriate storage directory (step 975). Finally the provider's notification preferences for the WPFileSend method are followed (step 976). For example, this step may involve placing a notification message element (211, FIG. 4) and a hyperlink to the file or files in the provider's notification report. Clicking this hyperlink will open the provider's word processor 958 and load the word processing file 952, as is the convention with files of specific MIME types submitted to a browser 50. In this way the provider can view the fully translated word processing file or files 952 in the optimal format without expending any human effort to receive, decompress, decrypt, or translate the file format.

Detailed Description Text (209):
The steps necessary for acknowledgment automation are shown in FIG. 1. When a consumer program 22 receives a communications object instance 35 from a provider program 11, the consumer program 22 executes the object's receipt methods. By including an SendAck receipt method in the communications object 35, the provider can cause an acknowledgment message 33 to be returned to the provider program 12. The SendAck method simply generates a message object instance (810, FIG. 17) that includes another SendAck receipt method and the recipient's database system ID (100, FIG. 3). When the acknowledgment message object 33 is received by the provider program 12, its SendAck receipt method 141 is executed. Being a

polymorphic method, the operation performed by the SendAck method 141 at the provider program 12 is different than at the consumer program 22. Referring again to FIG. 3, the SendAck method 141 first uses the system ID of its parent communications object 110 and the system ID of its originating recipient 120 to query the database for a acknowledgment association 121. It then changes the value of the AckFlag attribute to TRUE. The AckFlag attributes of all acknowledgment associations 121 are maintained in this way. Now all that is required to complete acknowledgment automation is for the provider program 12 to periodically execute a scheduled event 117. This event executes a AckMonitor method which queries for all acknowledgment associations 121 where AckDateTime is equal to or less than NOW and AckFlag is FALSE. Those instances meeting this criteria represent recipients 120 from whom acknowledgment messages have not been received in the alloted interval. The AckMonitor method could then take appropriate actions. For example, it could execute a designated notification method to notify the user, such as placing an entry in the user's notification report. Notification control is further discussed below. Alternatively, the AckMonitor method could automatically retransmit the appropriate communications object instance 110. Each time it did this it would increment the AckDateTime value of the acknowledgment association 121 by the appropriate Ackinterval. It would also increment the integer value of the RetryCount attribute by one. If the acknowledgment association 121 continued to fail the AckMonitor check, each subsequent retransmission would continue incrementing the RetryCount attribute until it equaled a RetryThreshold attribute value stored in the global preferences instance 103. At this point user notification could be triggered, as well as other appropriate actions designated by the provider, such as deletion or inactivation of the recipient instance 120.

Detailed Description Text (212):
Acknowledgment messages can still be used even when the distribution method uses the pull technique. This is accomplished identically to the above except for the following. First, the acknowledgment message object instance (810, FIG. 17) returned to the provider program 12 includes such additional data about the consumer as is necessary to create a recipient record 120. Second, if the recipient record 120 instance does not exist in the provider database 11, the SendAck method needs to create it, and also create the association 121 between the recipient 120, the communications object 110, and one or more methods 141, including an update method. This specialized use of an acknowledgment message object 110 is referred to as a registration message. Registration messages are important for three reasons. First, registration messages can be used to track communications object distribution and usage even when the provider does not have the capability to distribute updates using the push technique. An example is when an expensive, high-powered web server is used for high-volume distribution of a communications object, but an inexpensive personal computer and email account is used for tracking communications object acknowledgment messages. Second, registration messages can be used on an intermittent basis by only including the SendAck receipt method in selected communications object updates. This allows distribution statistics and other data to be gathered periodically rather than with every update. Third, if the acknowledgment message object 110 includes the e-mail address of the consumer, the resulting list of recipients 120 created by registration messages can allow the provider to convert the communications object update method from pull to push. Conversion between update methods is discussed further below.

Detailed Description Text (214):
Another example of a provider-assigned receipt method is registration. Certain communications objects such as service objects may explicitly wish to register themselves or their public methods within the consumer database 21. Object and method registration will be discussed further below.

Detailed Description Text (219):
One of the most distinguishing features of a communications object system is its ability to control the automatic updating of communications objects. Certain types

of communications objects, such as those designed for one-time data exchange operations, may not be persistent in the consumer database 21 and thus not require updating. However every communications object that will be persistent in the consumer database 21 needs to be updated when the provider makes changes to the object. Push-based updating is automated through the use of the update association rule (FIG. 10B) described above. Pull-based updating is accomplished through the use of update methods. As with any other object method, an update method may be a reference to a system method, an method carried internally in the object, or a call to a remote method stored on another computer accessible via communications network 3. A communications object may also be associated with multiple update methods.

Detailed Description Text (221):
When a communications object employs the pull technique of updating, an update method is used to control the update operation. Pull-type update methods can use any services available at the consumer program 22 to initiate an update. In a preferred embodiment shown in FIG. 1, an update method initiates a polling request from the consumer program 22 to a distribution server 32. For example, the consumer program 22 can issue a HTTP "Get" request to Web server using the "If-Modified-Since" parameter in the header. The date/time of the most recent existing communications object version in the consumer database 21 is supplied as the value for the "If-Modified-Since" parameter. This value is stored as the LastUpdate date attribute of the communications object (110, FIG. 3). If the date/time of the the communications object markup file 35 on the Web server has not changed, the Web server returns a response code indicating "no change", and the update method will schedule the next polling event. If the date of the file 35 stored on the Web server 32 is newer, the Web server returns the communications object markup file 35, and processing begins as shown in FIG. 16.

Detailed Description Text (222):
Referring again to FIG. 3, the triggering of update methods is typically controlled by a scheduled event instance 117 in the consumer program 22. As described above, this scheduled event instance 117 can be created by a receipt method executed when a communications object or object update 110 is received. It can also be scheduled or rescheduled by an update method triggered by a scheduled event instance 117. This combination of using receipt methods and update methods to control scheduled event instances 117 provides comprehensive control over update events. This is further augmented by the ability of receipt and update methods to use any data available to them within the communications object 110 or the consumer database 21 to make update decisions. For example, update events can be scheduled based on a specific periodic interval or specific date/time set by the provider. By the use of preference elements, the provider may also allow the consumer to choose an update interval or date/time, or to choose from within an update interval range or data/time range offered by the provider. The provider can also let the update method determine a random date/time within a periodic interval or date/time period. This last approach, commonly referred to as a "back off", can be useful for controlling server loads. For instance, a provider may publish a weekly newsletter on Friday afternoons at 2 p.m. Eastern Standard Time. By specifying that the receipt or update method schedule the update polling event for a random time within the next 3 hour period, the provider can efficiently distribute a very large volume of updates within that 3 hour period without overloading the server. Receipt or update methods can also use other logic or data to control update polling. This might include factors such as the total age of the communications object in the consumer database 21; the frequency with which the consumer has viewed or acted upon the communications object; costs or fees associated with an update; or other criteria. The specific algorithm or algorithms used to control update scheduling are not a limiting feature of the invention. The consumer may also wish to have the consumer program 22 dynamicaly reschedule update events depending on other factors such as the time of day, the interval since the program was last run, local or Internet network traffic levels, the consumer's own system activity level, other system or environment variables, disk space availability, or other factors. Update

methods can also be triggered manually by the consumer.

Detailed Description Text (226):
One communications object can be used to control the updating of other communications objects. For example, the receipt method for a composite communications object can trigger the updating of each of its component objects. To illustrate this, refer to FIG. 20 and the preceeding discussion of consumer distribution control using the pull technique. A composite communications object 900 can contain multiple page subscription element instances (853, FIG. 18) corresponding to its component communications objects 901. Each page subscription element instance can include an attribute for the current version value of the corresponding component object 901. This version value attribute can be maintained using a rule 140 that updates the version value of the page subscription element instance when the version value of the component communications object 901 changes. When the composite communications object 900 is updated, its receipt method can compare the version value in this attribute with the version value of the corresponding component object 901 currently stored in the consumer database 21. When the version value has changed, the update method of the corresponding component object 901 can be executed to update the component object. In this manner the component objects themselves do not need to be polled for updates. This same technique of "indirect updating" can be applied to any set of communications objects, where elements in one communications object are processed to trigger the updating of other related communications objects. In this way, for example, a single communications object at a web site could be used to check for updates on many additional communications objects on the web site or related web sites.

Detailed Description Text (227):
Another highly efficient update control technique, referred to as update query control, requires the use of database program operating on a distribution server 32. In a preferred embodiment, this can be accomplished using a distribution service object 1310 and a distribution partner server 1302. These will be further discussed in the distribution service object section below. With update query control, the communications object 110 controlling the updating need not contain any direct references to the specific communications objects or component objects being updated. Rather the controlling communications object 110 can contain one or more data exchange elements 143 and data exchange methods 141 which function as an update method. (Data exchange elements and data exchange methods will be further explained in the data exchange control section below.) The data exchange method 141 can first execute a query of the consumer database 21 for all communications objects which match the query critieria. For example, a composite communications object 900 could query for all its component communications objects 901. The query result includes the UID and current version value of each component object 901. The data exchange method then uses the result set to perform a second query of the distribution server 32. The distribution server 32 would return each component object 901 for which the version value on the distribution server 32 was greater. In this manner a single communications object could be used to very efficiently update thousands or even millions of communications objects stored on high performance database servers.

Detailed Description Text (228):
This process can be made even more efficient for the consumer by the maintenance of an index of provider UIDs and the communications objects 110 with which they are associated with on the distribution server 32. This process is further described in the directory service object section below.

Detailed Description Text (231):
One of the greatest advantages of a communications object system over other communications systems is the ability it gives information consumers to control notification about communications events. The fundamental reason for this is that within a communications object system all messages are transmitted and received as

some type of communications object. This allows messages to be "pre-processed" by the consumer program 22 or provider program 12 using data or methods from one or more communications objects already present in the consumer database 21 or provider database 11. This preprocessing allows these programs to do a large amount of the sorting, filtering, and notification work that otherwise would require human effort.

Detailed Description Text (240):
FIG. 4 illustrates two typical notification methods assigned to an element preference instance 221. A SendEMail method 224 causes each message element 211 associated with the notification element 202 to be sent as an e-mail message to an address or addresses specified by the consumer. Preferably, such an e-mail message would use as the start of its header a signifying string such as "Special Alert:", followed by the headline text value from the associated message element 211. The body of the message would then contain the body value from the associated message element 211. It could also contain the headline link value, body link value, and other status or navigational information, such as the name of the originating communications object, the name of the provider, or other actions taken. An AddToNotifyReport method 225 causes the headline of each associated message element 211 to appear in the consumer's notification report (630, FIG. 14). To set this trigger, the AddtoNotifyReport method adds a logical NotifyReportFlag attribute 223 to the element preference instance 221 and sets its value to TRUE. To display the notification report (630, FIG. 14), the consumer program 22 performs a query of the consumer database 21 for all message elements 211 associated with all element preference instances 221 where the NotifyReportFlag 223 value is TRUE. The actual content displayed in the report is determined by attributes of the consumer's global preferences (103, FIG. 3). The consumer may wish to see headlines only. In this case each headline can be displayed as a hyperlink. When selected, the hyperlink will display the message body and body links as a separate page. Alternatively, the consumer may wish to see all headlines, messages, and links in the notification report. Headlines may also be linked to other elements or methods, such as those used for data exchange. Headlines may also function as a hyperlink directly to another URL anywhere on the Internet. Another option is for the consumer to see communications objects for which there are new notifications displayed differently than other communications objects for which there are no new notifications. Notification reports may also be sorted according to the settings of the sort form (634, FIG. 14), or by using various toolbar buttons for common sorting or filtering options. For example, notification data could be sorted by communications object name, communications object nickname, date, folder, or notification priority. Different standard or custom notification reports can also be stored and presented as menu options or toolbar buttons. A example of a notification report sorted by date showing headlines only for communications objects which had new notifications is shown in FIG. 24. Each notification report entry can include a button for deleting the entry from the notification report immediately, or a checkbox for batch deletion, or both. In either case, when the notification report form is submitted, this button or checkbox causes the NotifyReportFlag attribute 223 of the corresponding preference element instance 221 to be set to FALSE. The format of a notification report is not a limiting feature of the invention.

Detailed Description Text (245):
Notification methods 141 can also be assigned to system events, and these too can be integrated into the same notification reports. For example, a system event can trigger notification that a provider is due to release a periodic communications object update, or a consumer that his/her consumer database 21 needs to be backed up.

Detailed Description Text (247):
The ability of a communications object system to automate common communications tasks is perhaps best exemplified by its ability to automate data exchanges between

consumers and providers. Typical examples include the exchange of contact information, demographic data, psychographic data, billing information, product registration information, customer service data, technical support data, transaction histories, stock feeds, news data, weather data, and so on. A communications object system is capable of automating the exchange of such data to a greater degree than any other existing communications system for five reasons. First, such data is already stored in a consumer database 21 in such a fashion as to be available for automated access and delivery. Second, such data is available in structured, typed formats that allow providers to easily specify the data they require. Thirdly, communications objects give providers the tool they need to transfer such data from the consumer back to the provider. Fourth, message objects and the architecture of the provider program 12 allow the provider to automate the processing of such data when it is received back at the provider. Fifth, the ability of the provider program 12 and consumer program 22 to automatically trigger events and respond to message objects means a multi-part data exchange transaction (such as a purchase and receipt acknowledgment) can be automated throughout.

Detailed Description Text (248):
The primary data structures involved with data exchange control are data exchange elements 143 and message objects 110, both described above. Any communications object method 141 involved with data exchange can be referred to as a data exchange method. Data exchange elements 143 in a communications object 110 can call a data exchange method 141. Data exchange methods 141 in the consumer program 22 can produce message objects that can be transmitted back to the originating provider program 12, or to any other program capable of processing the message object. Data exchange methods in the provider program 12 can also produce message objects that can be sent to any consumer program 22 containing the parent communications object. Like any communications object, message objects can contain any combination of components required to transport and process the data they contain. Data exchange methods that produce message objects can be fully automatic. For example, a receipt method can produce and transmit an acknowledgment registration message object, described above, with no consumer intervention. Data exchange methods that produce message objects can also obtain manual data input from the consumer or provider. In a preferred embodiment the mechanism for obtaining this input is an HTML form. Such input forms are produced and displayed by the provider program 12 or consumer program 22 in the same fashion as the forms produced for operation of the programs themselves. A typical input form is shown in FIG. 27. Each data field that accepts input on the form is an attribute of an element 143. Other text or graphics that appear on the form, as further instructions or directions to the user, are other elements either supplied by the provider or drawn from the consumer database 21 or provider database 11. Any communications object component stored in either of these databases may be included, subject to the consumers or provider's data access rules discussed below. The only difference between input forms produced by data exchange methods and standard program forms is that an data exchange input form is generated by and processed by a data exchange method. Alternatively, user input can be obtained through other user interface options including standard operating system functions such as dialog boxes and menus. The use of a graphical user interface will be specifically discussed below.

Detailed Description Text (249):
Besides input form and message object generation, data exchange control in a communications object system involves data type control, data persistence control, data access control, and data security control. Each of these must be considered from the standpoint of internal data, i.e. data within the provider database 11 or consumer database 21, and external data, i.e. data available elsewhere within the provider's or consumer's computing or network environment.

Detailed Description Text (250):
Data type control is required because providers need a way to specify the data they require in a specific data exchange transaction. The data type definition features

of a communications object system, as explained above in the data structure section, are ideally suited to this need. By creating a system-wide set of low-level composite type definitions, such as Name, Address, and Telephone, and then nesting these inside of progessively more comprehensive composite type definitions, such as BusinessCard or Resume, a hierarchy of standard data type definitions can be created that are available to all providers and consumers. This has two very significant advantages. First, as providers design input forms and methods for data exchange tasks, they can choose from among these standard data type definitions rather than needing to create their own composite data type definitions, saving considerable time and effort. Second, data type standardization means that consumers need only enter data once into each instance of each data type that pertains to them. For example, the consumer only needs to enter his/her name, addresses, telephone numbers, birthdate, and other personal data one time into the consumer database 21. From that point on all communications objects which need data of these types can access these data type instances. This saves the consumer data input time and also vastly reduces the potential for data input errors.

Detailed Description Text (252):
Providers can also create their own data type definitions and specify the use of these composite data type definitions in data exchange methods. When a provider-specific data type can be aggregated or calculated from other system standard data type definitions which are already present in the consumer database 21, the resulting element can be composed automatically by a data exchange method. When a provider-specific data type requires the input of new data from the consumer, an input form can be generated by the data exchange method. Once submitted, the data can also be saved as a element preference instance (147, FIG. 3) in the consumer database 21. The provider can then use the system ID of the type definition of this element to query for this element preference instance in future transactions. This allows a provider to dynamically generate and persistently store provider-specific data type definitions in the consumer database 21. A common example of such a data type might be a consumer's preference between a provider's selection of product colors, such as clothing or paint. Storing this data locally at the consumer database 21 means that it can easily be included in any future communications from the consumer. Additionally, such data can be shared among all communications objects or data exchange methods from that provider, as further explained below. Another key benefit is that this data can be easily and immediately edited by the customer should the customer's information or preferences change. Such changes can also be automatically transmitted back to the provider through the use of data association rules, discussed below.

Detailed Description Text (253):
As with any multiuser database system, shared access to data requires data access controls. This control should cover all common data operations such as creating, reading, writing, moving, and deleting data. In a communications object system, data access controls need to extend beyond human operators to communications objects, since these objects are essentially acting as "surrogates" for their respective providers. The key data structure involved with data access control is the rules class 140. Data access rules can monitor all forms of data access within the provider database 11 or consumer database 21 as well as external data in the provider or consumer's computing or network environment. For example, a typical rule governing access to communications object components or element preference instances might be that only other communications objects sharing the same database system ID (100, FIG. 3) can read, write, or delete such instances. This would prevent different providers from having access to each other's private data. This rule could be modified so that only communications objects sharing a group system ID (251, FIG. 6A), described above, could have access to such data. This would allow all communications objects created by employees of the same company, or within a company division, to access each other's communications object component or element preference instances. Data access rules can be system-wide, assigned by providers, or assigned by consumers. An example of a provider-assigned rule would

be restrictions on communications object forwarding, which will be further discussed below. An example of a consumer-assigned rule would be that designated personal data, such as household income, must be explicitly authorized by the consumer before it is transmitted in any data exchange. A stricter rule would state that more sensitive private data, such as credit card numbers, must be encrypted and require one or more passkeys for decryption prior to any data exchange. In order to protect their integrity, data access rules can also enforce the ability to add or change other data access rules, and also the hierarchy in which rules take precedence when two or more rules apply. Data access rules can also be selectively applied by the consumer to particular communications objects 110 or communications object groups such as folders 115 by creating associations between these and a data access rule 140. The application of rules to control data access within an active database is further discussed in the aforementioned Active Database Systems.

Detailed Description Text (255):
Data type, persistence, access, and security control can be applied to the exchange of data external to the provider database 11 or consumer database 21 in the same manner as internal data. Such external data falls into three general categories: system data, file data, and data available via external queries. System data includes system environment variables, configuration variables, and operating statistics. File data includes data available directly via operating system calls such as files, persistent system objects, or any other data stored directly in the user's local or network computing environment including removable storage devices mechanisms such as floppy disk drives, CD-ROM drives, or tape drives. Data available via external queries includes data stored in and available through other computer programs operating in the user's local or network computing environment, including application programs, database servers, naming or address servers, web servers, or any other type of server.

Detailed Description Text (257):
For external file data, data type control can be particularly useful. For example, personal computers running the MS-DOS or Microsoft Windows operating systems use a standard set of setup and initialization files including AUTOEXEC.BAT, CONFIG.SYS, WIN.INI, and SYSTEM.INI. Standard data type definitions can be created for elements that store information about each of these files, such as their name, size, date, and local directory path. A composite data type PCSetupFiles can also be created which included elements for each of these specific files. Providers can use these standard data type definitions to easily access the contents of these files for processing or data exchange. This access can be controlled by data access rules in the same way as internal data. This capability is particulary valuable for software or hardware technical support, where it can save both the provider and consumer considerable manual time and effort obtaining and exchanging this data.

Detailed Description Text (258):
A communications object system allows data persistence, access, and security control for external file data to operate at two levels. First are the standard file privileges granted to the user of the provider program 12 or consumer program 22 by the operating system or network administrator. Second are the rules 140 that can be enforced within the provider program 12 or consumer program 22 themselves. Data persistence control is particularly relevant to external file data. With the appropriate file creation privileges, data exchange methods can control the creation, modification, and deletion of external files on the user's computer system. These files can be used for many purposes, including the storage of message attachments, web helper files, log files, troubleshooting files, and files created by or intended for use by other software programs in the user's local or network computing environment. Access control and data security enforcement for these files, including encryption and authentication of individuals or communications objects requesting access, can be handled in the same manner as internal data. The ability to access and manage external file storage is particularly valuable in conjunction with the use of attachment elements. Attachment elements allow a

provider to store the specification for a file or files as a specific type of communications object element 143 which receives special processing during the communications object generation and transmission routine. This is shown as step 546 in FIG. 12. After the communications object itself is generated for transmission, any attachment element it contains is processed to determine the file, system object, or other attachment it specifies to attach to the transmission. Such attachments can be encoded in MIME, BinHex, UU encoding, or other attachment encoding format as described above. When the communications object bearing the attachment is received by the consumer program 22, the attachment is stored according to a corresponding receipt method. The attachment can be stored internally as an element 143 in the consumer database 21, or externally in the consumer's file system. File data exchange control can also be combined with notification control. For example, a message element (211, FIG. 4) including a hyperlink to the attachment can also be created for inclusion in a notification report (630, FIG. 13) by the consumer program 22. In this way communications object updates can serve as a powerful means of automatically distributing and indexing one or more external attachment files.

Detailed Description Text (259):
One of its most powerful forms of data exchange control in a communications object system is the ability to automate external data queries and the processing of query result sets. This is because it gives providers a tool to allow consumers quickly and easily set up automated queries against any type of data server maintained by the provider. These queries are easily set up because they can be composed using any data available in the consumer database 21 (subject to the consumer's data access rules, as explained above), so the consumer need only enter any new data required. The queries are easily automated because the data exchange method that executes them can create its own scheduled event instances (117, FIG. 3) to execute future instances of the query. External query control can also be combined with notification control to automate notification depending on the query results. For example, a data exchange method that executes a data query for a stock price can notify the consumer if the new price is a certain dollar amount or percentage amount changed from the previous price.

Detailed Description Text (260):
Data type control is especially useful with external queries. This is because the use of standardized data query languages such as Structured Query Language (SQL) makes it easier for providers to create or consumers to modify routine data queries. SQL and other approaches to standardized data query languages are discussed generally in R. G. G. Cattell, Object Data Management--Object-Oriented and Extended Relational Database Systems (1994), which is incorporated herein by reference. In a communications object system, the specifications for a structured query can be stored as a special type of communications object element 143 called a query element. Query elements receive special processing during the communications object generation and transmission routine. This is shown as step 545 in FIG. 12. After the communications object itself is generated for transmission, it is tested for any query elements. For each query element it contains, the data exchange method associated with the query element is executed to perform the query. This could be a query against the provider database 11, against another local application acting as a server, against a network database server, against a web server, or against any other server capable of query processing. When the query result set is returned, the data exchange method determines what further steps to take. These may include appending the data to the communications object transmission as a file attachment, creating and appending a message object, or otherwise modifying the communications object or its encoding or transmission. Query elements thus provide a powerful extension to a provider's ability to control and customize communications object distribution.

Detailed Description Text (261):
Data persistence, access, and security controls all apply to external data queries

as well. Again, a communications object system allows these to be implemented at two levels. At one level, these can be the same controls that apply to the human operator of the programs 12, 22. For example, the user's ability to read, write, or create new records in a database server can be governed by a user ID and login password controlled by a system administrator. The programs 12, 22 can simply require the same information to be entered manually. Alternatively, the programs 12, 22 could store this information as global preferences that it can submit automatically as part of executing data exchange methods. The programs 12, 22 can then implement their own layer of internal security. This can include the use of system-wide login names and passwords, the implementation of rules 140 controlling data access, and the encryption of sensitive data, all as described above. Data access and security control is particularly useful when data exchange methods employing queries are executed by the consumer program 22 on the behalf of the consumer. Using such controls, a provider is able to select the subset of consumers on a communications network 3 such as the Internet who will have access of some kind to one or more databases or database servers operated by the provider. This control is useful when the provider wishes to charge access fees for the data, to protect the data for competitive or security reasons, or to monitor or track access to the data.

Detailed Description Text (263):
As explained above, when the provider program 12 and consumer program 22 are combined, the same facilities for processing communications objects on behalf of the consumer are available for processing message objects on behalf of the provider. For example, message objects 110 can contain or produce message elements (211, FIG. 4) for inclusion in a notification report (630, FIG. 13). Because notification reports are produced by queries against the databases 11, 21, these message elements can be sorted by any criteria desired by the provider. For example, they can be sorted by their parent communications object (110. FIG. 3), by related notification element (201, FIG. 4), by the type of action required by the provider, or by any other element contained in the message object which produced them. Within the notification report, message elements can be hyperlinked to other data exchange methods to control further processing of the message object data. As with data exchange methods in the consumer program 22, this can include the generation of input forms for gathering additional input from the provider and the generation of message objects that can be returned to the consumer or to other data servers.

Detailed Description Text (265):
Whether called up manually by the consumer or automatically by an API call, the appropriate page (142, FIG. 3) within the communications object 35 would display the various technical support options offered by the company (step 1110). FIG. 26 is an illustration of how such a page might typically appear. Each data exchange element 143 representing a support option appears as a hyperlink command or button. When activated by the consumer, this command calls the data exchange method 141 associated with the data exchange element 143. The data exchange method 141 then generates an input form for gathering additional data from the consumer (step 1111). FIG. 27 illustrates how such an input form might typically appear. As described above, an input form can display fields for manual input from the consumer which are attributes of a data exchange element. These include checkboxes, radio buttons, drop-down lists, and text input fields. Text input fields can be used for free-form text input such as writing technical support questions in a manner similar to writing e-mail messages. The input form can also display data already present as elements in the consumer database 21 for confirmation and authorization by the consumer. When the consumer submits the input form, the data exchange method 141 processes the form input in the same manner that system methods process program form input as described above (step 1112). This includes any error detection routines, which may display additional messages or input forms. Once the form data is validated, the data exchange method 141 produces a message object instance 1115. This message object instance may include internally or as

attachments any of the different types of exchange data discussed in this section. For example, it could include system data such as the time and day, computer type, CPU type, and RAM available. It could also include file data such as the consumer's AUTOEXEC.BAT or the initialization file for the supported software program 1102. It could also include data from a data query such as a error log report produced via an API call to the supported software program 1102. The automated inclusion of this data not only saves a great deal of manual effort on the part of the consumer, but elimates manual data entry or attachment errors. Finally, the data exchange method 141 transmits the message object instance 1115 to the provider program 12 (step 1117).

Detailed Description Text (270):
Referring to FIG. 3, the preceeding discussion of data exchange has focused primarily on data exchange events initiated either manually by the consumer or automatically by scheduled events 117. Either one of these techniques can be employed to either pull new data to the consumer or push data from the consumer to the provider. However data exchange events can also be triggered automatically by data exchange rules 140. Perhaps the most common example is the update association rule discussed above. The update association rule (FIG. 10B) ensures that changes made within the provider database 11 are distributed to all associated recipients 120 via all associated communications objects 110. When the provider and consumer programs 12, 22 and databases 11, 21 are combined, a corresponding data exchange rule 140 can be employed by a consumer communications object 110 to monitor changes to one or more provider elements 143. Such a data exchange rule creates an association between an element 143, a communications object 110, and a data exchange method 141. When the version value of the element 143 changes, the rule 140 is invoked which calls the data exchange method 141 of the communications object 110. The data exchange method 141 can then produce a message object 110 to transmit the changed data back to the provider program 12. Alternatively, a message object or another type of structured message can be transmitted via any available communications network to any other address the provider designates. In this fashion a communications object consumer who is also a provider can automatically update all of his/her communications object relationships associated with a particular element 143 just by editing the element 143. A common universal example is change-of-address notifications. With any other communications object network, such as postal systems, telephone systems, fax systems, or e-mail systems, a change of address involves significant human labor on the part of the party whose address is changing to notify all his/her communications partners. For their part, each communications partner must update their own records when a change-of-address notification is received. When data exchange rules 140 are employed in the present invention, every communications partner is updated automatically whenever relevant communications data elements change. This includes both partners who are recipients 120 of the user's own communications objects 110, as well as partners from whom the user has received a communications object 110 employing data exchange rules. As the number of users of a communications object system grows, the overall human labor savings involved in this automatic two-way updating of contact data can become very significant.

Detailed Description Text (271):
A specific application of data exchange rules is registration updates, discussed in the update control section above. A registration update is a message object returned by a consumer program 22 to a provider program 12 or distribution server 32 in order to modify a recipient instance 120 and/or other elements associated with a communications object 110. Typically any element 143 in the consumer database 21 that is part of the registration data maintained in the provider database 11 will be monitored by a data exchange rule 140 contained by the communications object 110. Any changes to these elements 143 will result in a message object being produced that produces an update in the corresponding recipient instance 120 or other element 143 in the provider database 11.

Detailed Description Text (272):
Another example of data exchange rules is version monitoring. Version monitoring can be a more efficient updating technique when the message object traffic volume produced by a communications object 110 is more frequent than changes to the communications object 110 itself. With version monitoring, changes to a communications object 110 are neither explicitly pushed by the provider program 12 nor regularly pulled by the consumer program 22. Instead each message object passed between the programs 12, 22 contains the most recent version value of the parent communications object 110. A version monitoring rule 140 operating in either or both programs 12, 22 compares this version value in the message object with the current version of the communications object 110. Whenever a change is detected, the version monitoring rule 140 executes the update method 141 of the communications object 110 to update the current version in the consumer database 21. A specific example of version monitoring is shown in FIG. 37, explained in the payment service object section below. Version monitoring rules 141 can also be implemented at distribution servers 32 to add efficiency to the update polling process. This is discussed further in the data archiving control section, below.

Detailed Description Text (276):
A communications object system overcomes all of these disadvantages. Using a communications object 110 employing data exchange methods 141 to control the relationship, the provider can first gather most or all of the consumer's preference data automatically. This can be controlled by rules 140 imposed by the consumer. The provider's communications object 110 itself can create the necessary ID for the consumer using the system ID (100, FIG. 3) of the consumer database 21 or a derivative thereof. The consumer is not required to give a password manually because the provider's communications object 110 can communications with the consumer program 22 to establish the consumer's identity (the consumer's own indentification and preference elements 143 stay safely within the consumers own computing environment). The provider's communications object is able to automatically download and selectively notify the consumer of new personalized content as described above. Finally, any changes to identification or preference elements 143 are made once locally and are transmitted automatically to all such web provider relationships.

Detailed Description Text (278):
A communications object system overcomes these limitations by replacing the cookie with a communications object 110 from the provider. In fact such an improvement can be made under the existing HTTP protocol if a communications object exchange is initiated manually by the consumer during a browsing session by clicking on a hyperlink representing a communications object 110 on a web page presented by the web server. The resulting download of a communications object 110 can trigger a data exchange receipt method 141 which automatically transmits back to the web server any necessary data elements 143 from the consumer database 21. This can be controlled by rules 140 imposed by the consumer. The web server can then prepare and return customized content for the consumer program 22 to display to the browser. Alternatively, the web server can return another communications object 110 to repeat the information interchange process. In contrast to cookies, the consumer can be completely in control of this process. The consumer can view the elements 143 of the relevant communications object; edit those elements 143 which involve consumer preferences; and apply rules 140 governing data access, data security, and data logging by the communications object. These improvements can bring rich, automated new forms of web content personalization with none of the disadvantages of cookies.

Detailed Description Text (280):
A communications object system overcomes these limitations by allowing providers and users to control communications relationships at the level of individual communications objects 110. Any number of providers and consumers can take part, each with access to the full range of data exchange control provided by

communications objects 110 and data exchange methods 141. Thus a communications object system can be used to provide any number of of personalized content delivery services, rather than the limited number offered by one particular application. For example, data query control can be used to submit an HTTP request to any web server. This query request can include a unique identifier for the consumer such as the consumer's UID or a derivative thereof. This query can also include any new or updated data elements 143 in consumer database 21 that pertain to the provider and are not already present or current at the web server. The elements 143 or communications objects 110 returned by this query can offer completely customized news reports, weather reports, product brochures, advertisements, stock quotes, real estate listings, classified listings, Internet searches, health care advice, or any other current personalized information desired by the consumer. The specific data type is not a limiting feature of the invention. The consumer can also control notification options for this information down to the level of elements 143. The information can also be hyperlinked to data exchange elements 143 within the relevant communications object for direct, automated feedback by the consumer.

Detailed Description Text (287):
This approach requires that all address resolution logic be present in one of two places: in the DNS protocol, or in the address resolution methods available at remote hosts. With a communications object system, the address resolution logic first resides in the link method 141, which can in turn call any other communications object method 141 or partner server method 141 as described in the section on communications object methods above. This means that by using a UID, URL, name, or other attribute or combination of attributes supplied in a link element 143, the link method 141 can first search the local databases 11, 21 within the programs 12, 22 for the specified communications object or objects. If the communications object is not found locally, the link method 141 can then query one or more distribution servers 32 where the communications object is likely to be stored, such as LAN or WAN distribution servers. These distribution servers 32 may be represented by distribution service objects 1310, which will be further discussed below. If the target communications object 110 is not found on a distribution server 32, then the link method 141 can process the attributes of the link elements 143 to determine the next most efficient means of retreiving the communications object. For example, if a URL is available, the link method 141 could process this. If a URL is not present but a UID is available, the link method 141 could automatically use a UID resolution service. If a UID is not present but a name is available, the link method 141 can use a name resolution service. UID or name resolution services can operate similarly to the Domain Naming Service (DNS) for the Internet, or to the PURL naming service cited above. Additionally, the name resolution service could incorporate features under consideration by the World Wide Web Consortium (W3C) for Uniform Resource Identifiers (URIs) and Uniform Resource Name (URNs). These systems are discussed generally by the W3C staff at the W3C World Wide Web site at http://www.w3.org/pub/WWW/Addressing/.

Detailed Description Text (289):
When the provider program 12 and consumer program 22 are combined, a communications object system can also automate the exchange of communications objects 110 between two providers. This is a common need analogous to the exchange of business cards between individuals. Although only one business card need be exchanged to create a communications relationship, a two-way exchange provides the greatest number of communications options in both directions. This synchronization can be accomplished by the use of a special method 141 called an autoexchange method. An autoexchange method 141 operates as both a receipt method 141 and a data exchange method 141. The the data structures involved are illustrated in FIG. 3. When a provider program 12 first receives a communications object 110 with an autoexchange method 141, it first compares the database system ID of the communications object 110 with those of its recipient instances 120. If the database system ID is already present, the relationship is already established, and the autoexchange method is ignored. If not, the autoexchange method 141 is executed according to the receiving provider's

preferences. Such preferences can be specified using the provider preferences form (316, FIG. 9) and stored in the global preferences instance 103. One preference may be to automatically generate and transmit a default communications object 110 instance back to the first provider. Another preference may be to generate a message element (211, FIG. 4) for use with the notification report (630, FIG. 13) to notify the receiving provider of the autoexchange request. The headline of the message element (211, FIG. 4) could be linked to the create new recipient form (311, FIG. 9). This form would serve as the input form for the autoexchange method 141. The received communications object 110 could supply the attributes necessary for the recipient instance 120 on this form. Therefore the receiving provider need only select the preferences for the communications object or objects 110 to be returned to the first provider. When this form is submitted, the autoexchange method will trigger the distribution of this communications object as described above.

Detailed Description Text (292):
A special case of communications object exchange is when a consumer wishes to send a provider's communications object in the consumer's database 21 to another consumer. This is the real-world equivalent of a businessperson needing to share the contact information for a customer with a business associate, or a mail order customer wanting to tell a friend how to subscribe to a mail order catalog. A communications object system can accomplish this using forwarding methods. A forwarding method 141 allows the user of one consumer program 22 to transmit another provider's communications object via the push technique to another consumer program 22.

Detailed Description Text (293):
Execution of a forwarding method typically displays the forward form (635, FIG. 13). The forward form allows the user to manually input the e-mail address, encoding data, and any other data necessary to forward the communications object 110 to the recipient. Alternatively, the user can select the recipient from a list of recipients (120, FIG. 3) already present in the consumer database 21. In this case, the recipient instance (120, FIG. 3) contains the necessary information to address, encode, and transmit the communications object. When this form is submitted, the forwarding method generates a message object 110 and transmits it to the recipient. The contents of this message object can vary with the type of forwarding desired. An anonymous forwarding or "redirect" operation would simply send the communications object 110 itself, exactly as if it had been pushed from the original provider. This communications object would be received at the recipient's consumer program 22 just like any other communications object 110. A signed forwarding operation would include a forwarding element 143. A forwarding element is a recipient instance 120 representing to the forwarding consumer, e.g. their system ID, name, e-mail address, and so on. This information could be displayed by the notification method at the receiving consumer program 22. A signed forwarding operation could also include a message element (211, FIG. 4). In this case the forwarding consumer could control the headline and message displayed by the notification method at the receiving consumer program 22. A signed forwarding operation could also include copies of the forwarding consumers element preference instances (147, FIG. 3). In this case the receiving consumer would be able to choose whether to adopt these preferences or start with the default preferences of the original communications object 110. Any of these forwarding operations could also be authenticated, which would add an element 143 with a digital signature verifying the identity of the forwarder. Authentication is described in the encoding control section above, and authentication service objects are further described below.

Detailed Description Text (300):
Methods 141 used to transfer communications objects between consumers are referred to as a transfer methods. Just as a move operation on a computer file is often implemented by the operating system as a copy operation followed by a delete

operation on the original file, a communications object transfer method may be carried out as a forwarding method followed by a termination method. In addition, just as most computer operating systems first confirm the successful copy of the file to the new location before deleting the file in the old location, a communications object system should preferably allow consumers to confirm the successful transfer of the transferred communications object to the new consumer before deleting the communications object from the transferring consumer. This can be accomplished by using of message objects 110 as described above in the receipt and acknowledgment control section. Specifically, as illustrated by the data structures in FIG. 3, a transfer method 141 generates a message object 110 to the receiving consumer in the same manner as a forwarding method. The primary difference is that the message object includes a transfer receipt method 141 that adds a transfer rule 140 and an associated scheduled event 117 to the receiving consumer's consumer database 21. The transfer rule 140 is associated with the transferred communications object 110 such that when an update to the communications object 110 is received, the rule 140 is triggered. The rule 140 then executes the transfer method 141 which first deletes the associated scheduled event 117. The transfer method 141 then produces a message object 110 and transmits it back to the transferring consumer program 22. This message object invokes the originating transfer method 141 which then deletes the associated communications object 110. Optionally, the transfer method 141 could also produce a message element (211, FIG. 4) to provide notification to the transferring consumer that the transfer is complete. If an update to the communications object 110 in the receiving consumer program 22 is not received before the scheduled event 117 is triggered, it means an error condition likely exists. In this case the scheduled event 117 can produce a message element (211, FIG. 4) to provide notification to the receiving consumer. The scheduled event 117 can also execute the transfer method 141 which produces a message object 110 and transmits it back to the transferring consumer program 22. This message object invokes the originating transfer method 141. This transfer method can then either attempt a retransfer of the communications object 110, or produce a message element (211, FIG. 4) to provide notification to the transferring consumer, or both. The options for automatic error correction are more fully described in the receipt and acknowledgment control section above.

Detailed Description Text (301):
As with forwarding control, transfer control can also be exerted by the original provider using transfer rules 140 and transfer methods 141 in the communications object 110. Transfer rules and transfer methods are a particularly powerful means of data exchange control because they can accomplish automatic data exchange events involving the provider, the transferring consumer, and the receiving consumer all in one. An example is the transfer of ownership of a real world object, such as an automobile. A real-world rule applies to such a transfer, namely that the selling consumer must notify the automobile licensing authority, and the buying consumer must apply for a new title from the licensing authority. In this case the licensing authority would be the provider of a communications object 110 representing an automobile title. The selling consumer would have obtained the communications object 110 when he/she purchased the automobile. Using data exchange methods as explained above, the licensing authority would have used the communications object 110 to obtain the necessary elements 143 from the consumer required by law to register the vehicle. The licensing authority could then use updates to the communications object 110 to communicate with the consumer about the license, such as sending notifications about annual license renewals. (Payment for such license renewals can also be automated by data exchange methods in the communications object 110, as further described below.) When the time came to transfer the automobile title, the selling consumer would invoke the transfer method 141 in the communications object 110. The transfer method 141 would first generate an input form requesting the necessary data about the buying consumer and transaction details. (If a communications object 110 representing the buying consumer was also present, an association with such object 110 could be used to provide such data.)

The transfer method 141 would then produce two message objects 110. The first message object would be transmitted to the licensing authority, containing the necessary elements 143 to automatically register the sale. The second message object would be transmitted to the buying consumer. This would include the forwarded communications object 110 representing the title. A transfer rule 140 would also determine which element preference instances 147 must be transferred with the communications object 110. For example, the Vehicle Identification Number (VIN) must be transferred with the title; a new VIN may not be specified by the buyer. The transfer method 141 would also add a rule 140 to the selling consumer's database 21 requiring that affirmative acknowledgment message objects needed to be received from both the licensing authority and the buying consumer before the communications object 110 representing the title will be deleted. The transfer method 141 could also create a scheduled event 117 that checked for the receipt of these message objects after a specified interval.

Detailed Description Text (302):
On the buying consumer's part, the message object 110 received with the transferred communications object would invoke a transfer method 141. This transfer method 141 would first add a transfer rule 140 to monitor updates to the communications object 110 to ensure that it was operating properly. The transfer method 141 could also produce a scheduled event 117 associated with the transfer rule 140. This scheduled event 117 would check for the receipt of a reply message object from the licensing authority. Next the transfer method 141 would produce an input form requesting confirmation of the purchase and application for the new title by the buying consumer. It is signficant to point out that the buying consumer should not need to enter a single piece of data other than authentication or confirmation information. All such data is already present either in the communications object 110 or the consumer database 21. The transfer method method 141 would then produce a message object 110 to be transmitted to the licensing authority with the title application. When the licensing authority returned a acknowledgment message object 110 to the buying consumer, it would trigger the transfer rule 140. The transfer rule 140 would execute the transfer method 141. The transfer method 141 would first delete the transfer rule 140 and its associated scheduled event 117. The transfer method 141 would then produce another acknowledgment message object 110 to return to the selling consumer. When this acknowledgment message object was received, the same sequence of events would take place. The transfer rule 140 would execute the transfer method 141, which would first delete the transfer rule 140 and its associated scheduled event 117. It would then delete its associated communications object 110 and then delete itself. This would complete the transfer, with all parties assured of verified data delivery to the others.

Detailed Description Text (306):
However, in many communications systems, it is often very difficult for a consumer to terminate a communications relationship. An example is the difficulty many consumers have in being removed from direct mailing lists or telephone solicitation lists. With the communications system of the present invention, consumers have complete and immediate control over any communications object relationship. From the consumers perspective, this is accomplished simply by deleting the corresponding communications object 110 using the delete object form (623, FIG. 13). If a termination method 141 (FIG. 3) is present in the communications object 110, the delete object form calls this method; otherwise, it calls a default system termination method 141. A system termination method 141 would simply delete the communications object 110 from the consumer database 21. Optionally the system termination method 141 can replace the association between the communications object 110 and the database 100 with an association between the communications object 110 and a special "trash" folder 115. This permits the user to change his/her mind and recover the terminated communications object 110 at a later date by reversing this operation. A similar technique can also be used just to temporarily suspend or deactivate a communications object 110, which could later be reactivated by reversing the operation.

Detailed Description Text (308):
If a communications object 110 is updated via the pull technique, it is not necessary for a message object 110 to be returned to the provider program 12. Deletion of the communications object 110 at the consumer program 22 will terminate the relationship. However, the provider may still wish to employ a termination method 141 to receive overt notification of this event. Furthermore, regardless of the update technique used for the communications object 110, the provider may wish to employ a termination method 141 as a data exchange method. A common reason for doing this would be to ask the consumer why he/she is terminating the communications relationship. In this case the termination method 141 generates an input form where the consumer could indicate his/her reasons by selecting from checkboxes, radio buttons, or drop-down lists, entering text into a text box, or any other means of data input. When this input form is submitted, the termination method 141 generates a message object 110 and transmits it back to the provider program 12. At the provider program 12 a receipt method can automatically compile the consumer's input by storing it as elements 143, incrementing counters within elements 143, storing it in an external database, or any other data processing method. In this fashion the provider could periodically review aggregate statistics and/or direct textual feedback from consumers about why they terminated the communications relationships.

Detailed Description Text (310):
A communications object system provides a unique mechanism for enforcing the termination of a communications relationship. In the course of a communications relationship, a provider may have obtained a consumer's e-mail address. If so, the provider would have the ability to send the consumer new communications objects even after the consumer has already terminated the communications relationship. Although the consumer is able to delete these new communications objects with a single action, a large number of providers taking this action still requires significant effort and irritation on the part of the consumer. This is essentially the electronic equivalent of "junk mail". To prevent this, the consumer database 21 can track all or selected terminated communications objects 110. Such a record is commonly referred to as a "kill file". This is accomplished using a termination rule 140. First, the termination rule 140 requires that any termination method 141 executed in the consumer program 22 replace the association between the communications object 110 and the database 100 with an association between the communications object 110 and a designated "kill folder" instance 115. The termination rule 140 may also make this an optional checkbox on any input form generated by the termination method 141. Second, the termination rule 140 is triggered by the receipt of any new communications object 110. The termination rule 140 executes a termination method 141 that compares the UID of the new communications object 110 with the UIDs of all communications objects 110 associated with the kill folder. If there is a match, the termination method 141 can take whatever action is specified by the consumer. Typically, this will be to delete the communications object 110 without notification to the consumer. Alternatively, the termination method 141 could compare only the provider ID (the system ID of the provider database 100) with the provider ID of all communications objects 110 associated with the kill folder. This would detect any communications object 110 produced by a particular provider, not just a specific communications object 110. Another option is for the termination method 141 to track the number of attempted transmissions for any particular communications object 110 by incrementing an integer value attribute of the association between the communications object 110 and the kill folder 115. When this integer value reached a threshold, the termination method executes a notification method notifying the consumer, who may then take appropriate action.

Detailed Description Text (316):
By functioning as active databases, the provider program 12 and consumer program 22 can control the archiving of the data they store. This is a very useful capability

for many communications functions. First, many providers and consumers frequently wish to refer to past communications data. When stored in electronic format, this data is also computer searchable, which is another key advantage. Additionally, archiving data can be useful for error correction, as explained below. As shown in FIG. 3, data archive control is achieved primarily through the use of archive attributes, archive rules 140 and archive methods 141. The application of data archiving rules 140 in both the provider program 12 and consumer program 22 is explained above and in steps 735-737 of FIG. 15. These examples show how an archiving attribute and rule are used to control the number of previous versions of a communications object that will be archived. Archiving rules also allow control of archiving using time intervals, data size parameters, by the presence or absence of element preferences 147, and other parameters.

Detailed Description Text (317):
It is particularly useful for both providers and consumers to be able to control archiving at both a global database level and a communications object level. It can also be useful to control archiving at the element level. Global archive control is accomplished by storing communications object archive preferences as attributes of global preferences 103 and applying system archiving rules 140 and system archiving methods 141. Individual communications object archive control is achieved by storing archive preferences as attributes of the communications object preferences element 127 and applying archive rules 140 and archive methods 141 associated with the communications object 110. Element archive control is accomplished by storing archive preferences as attributes of the element 143 and applying archive rules 140 and archive methods 141 associated with the element 143. These three levels can also be intermatched. For example, if a communications object included a archive preference attribute in its preference element 127, but had no archive method 141, then the global system archive method 141 uses the local archive preference attribute.

Detailed Description Text (319):
In general, providers only need to control archiving in the provider database 11 and consumers only need to control archiving in the consumer database 21. However, archiving control can also be combined with distribution control and data exchange control as a way to ensure the versions of a communications object in a provider database 11 and a consumer database 21 stay synchronized. This is another aspect of version monitoring, discussed above. Version monitoring is desirable because it is possible for the consumer to miss versions of a communications object instance 110. For example, if the communications object instance 110 is distributed via the push technique, communications network errors may cause the transmission to fail. If the communications object instance 110 is distributed via the pull technique, communications network errors or distribution server errors may also prevent an update from occuring, although the polling retry process can frequently correct this. A more likely scenario is that either the consumer computer 2 or the consumer program 22 is not operated by the consumer during one complete version update cycle by the provider. For example, if the provider updates the communications object once per day, but the consumer does not operate the consumer program 22 for a week, the consumer may have missed six communications object versions. Finally, a communications object version could become corrupted in the consumer database 21.

Detailed Description Text (320):
If a uniform version value algorithm is used throughout the communications object system to increment the value of version attributes of communications object instances 110 or its components, recovery of lost or missing versions is straightforward. When the push technique of updating is used, recovery is accomplished using a version monitoring rule 140 and version monitoring method 141 at the consumer program 22. These can be implemented by the provider or the consumer. The version monitoring method 141 operates as a specialized data exchange method 141, explained above. The version monitoring rule 140 is associated with the communications object 110 so it is triggered each time an update is received. The

version monitoring rule 140 executes the version monitoring method 141 which compares the version value of the update received with version value of the most recent communications object 110 stored in the consumer database 21. If the version value algorithm indicates that a version value is missing in the sequence, the version monitoring method 141 generates a message object 110 containing a data exchange element 143 specifying the missing version values and the system ID of the consumer. The version monitoring method 141 then transmits the message object back to the provider program 12. Here, the message object executes a version monitoring receipt method 141. The version monitoring method 141 then executes the communications object instance generation and transmission routine (FIG. 12) for the specified missing version communications object versions and the specified recipient 120. When these new instances are received by the consumer program 22, synchronization is restored.

Detailed Description Text (322):
A second technique is to store the date and network address of previous versions in the communications object 110 itself. This approach has the advantage of allowing any number of version naming algorithms to be used. This can be done with three minor modifications to the communications object instance generation and transmission routine (FIG. 12). The first modification provides that each time a new version of a communications object instance 110 is generated, the version naming algorithm is used to generate a unique network address name for this version. This unique network address name together with the version value and current date and time is stored as a archive composite type element 143 contained in the communications object 110. This element is itself maintained by an archive rule 140 such that only n instances, or only instances newer than X interval, are stored. The second modification is the same as described above, i.e., the communications object instance generation and transmission routine uses the unique network address name to rename a copy of the previous communications object version instance 35 stored on the local or network file system. Alternatively it could regenerate the previous instance of the communications object instance 35 it were missing. Finally, at the conclusion of the routine, both the current and renamed previous version of the communications object instances 35 are be transferred to the distribution server 32. Now when the new communications object instance 35 is received at a consumer program 22, the version monitoring rule 140 executes a version monitoring method 141 as above. The version monitoring method 141 compares the version value attributes of the archive elements 143 contained in the updated communications object instance 35 with those in the consumer database 21. If any are missing, the version monitoring method 141 uses the network address stored in the archive element 143 to execute a polling operation for each missing communications object from the distribution server 32, restoring synchronization.

Detailed Description Text (325):
A communications object system can also automate a different form of archiving commonly required of any software program using a database. This is the storage of backup copies of the provider database 11 or consumer database 21 in order to prevent data loss from corruption, hardware failures, or other problems. In this case an archive method is functioning as a data exchange method 141. Backup control can be accomplished using backup preferences stored as attributes of global preferences 103, or as special backup elements 143, together with backup methods 141. Backups can be performed manually, or automated using scheduled events instances 117. Backup services can also be automated via a data exchange service object, which will be further discussed below.

Detailed Description Text (327):
Reports are typically one of the most valuable functions of any database system. In a communications object system, reports have particular value because they can be delivered automatically using the same system about which they are reporting. In this sense reporting control is simply a specialized case of data exchange control. Reporting control is especially useful to providers because it can give them

valuable statistics and feedback about the communications needs and behaviors of their audience. Reporting control can also be used by the operators of a communications object system as the basis for billing and licensing, much as telephone usage reports are the basis for billing telephone system customers.

Detailed Description Text (328):
As shown in FIG. 3, reports can be compiled on any class or group of classes in a database 100. These include specific communications object components 140, 141, 142, 143, and 144; individual communications objects 110; folders 115 or other groups of communications objects 110; recipients 120; and events 116, scheduled events 117, and logged events 118. Reports which are locally useful to the provider or consumer can be defined and executed using report instances 105. These reports are activated using the other reports form (640, FIG. 13). Report instances 105 are in essence a special case of the overall reporting capabilities of the system. If a report does not exist as a report element 105, a report is generated using a reporting element 143 and a reporting method 141. Reporting methods 141 function as a special case of data exchange methods 141, discussed above. As with data exchange methods, reporting methods are governed by reporting rules 140. Reporting rules function as a special case of data access rules 140, discussed above.

Detailed Description Text (329):
A unique feature of a communications object system is its ability to produce reports about the metadata it uses to control communications. This can happen at both the communications object level and at the element level. A specific example is communications object subscription reports. Referring to the example in the notification section above and illustrated in FIG. 4, in a communications object 110 a provider may create topic elements 201 from which a consumer can choose their notification preferences for topics of interest. These preferences are stored in a element preferences instance 221. By including a reporting rule 140 and a reporting method 141 in the communications object 110, the provider can be updated on any changes to user's preference element instances 221. The reporting rule 140 operates as a data exchange rule monitoring the element preference instances 221. When an element preference instance 221 is created or edited, the reporting rule 140 triggers the reporting method 141 which generates a message object 110. This message object contains the changed attributes of the preference element instances 221. The reporting method 141 next transmits the message object back to the provider program 12. When it is received, the message object calls another reporting method 141 to process the reported data, for example by creating or updating associations to the recipient instance 120. The provider is now able to run reports 105 against this data in the provider database 11 to produce statistics about communications object usage. These can include the total subscribership to any communications object 110 as well as the total subscribership to any particular topic element 201. In this way the provider can see the precise communications needs of the provider's audience. This feedback loop also allows providers to further refine communications topics and messages to better meet the needs of this audience. Alternatively, the first reporting method 141 can transmit a message object 110 or another form of structured message to another reporting server. Reports can be generated in the same fashion at this reporting server. This process can also be automated using reporting service objects, which will be further discussed below.

Detailed Description Text (330):
In addition to control of metadata, reports can also monitor communications activity. Continuing the example above, a communications object 110 can include a reporting rule 140 which monitors access to any message element instance (211, FIG. 4) contained in the communications object 110. When the message element is read from the consumer database 21 for the first time (i.e. read by the consumer), the reporting rule 140 is triggered. This executes a method 141 that creates a logged event instance 118 recording the UID of the communications object 110 and message element (211, FIG. 4). The communications object 110 can also include a receipt

method 141 that creates a scheduled event instance 117. This scheduled event instance will periodically execute a reporting method 141 which queries the consumer database 21 for new logged event instances 118 matching the UID of the communications object 110 and the appropriate event type. This reporting method 141 can then produce a message object reporting the results as explained above. Any data present in the queue of logged event instances 118 can be reported on in this fashion. As with other forms of data exchange, protection of both provider and consumer security is achievable through the use of reporting rules 140. Such rules can limit reporting access, for example, to logged event instances 118 which matched a particular communications object UID or a provider group UID.

Detailed Description Text (334):
Any given service object (815, FIG. 17) may provide services to providers, consumers, or both. Service objects that offer services to both providers and consumers are called polymorphic. Polymorphic service objects are particularly useful in a communications object system because many of the same services are required by both partners to a communications relationship, each in a different form depending on whether the partner is the provider or consumer. Such services typically fall into three categories: editing or searching databases, encoding or decoding communications, and automating transactions with third parties. An example of the first category is a directory service object, which permits providers to place or update listings in a directory service and permits consumers to automate searches of the same directory service for a specific provider. An example of the second category is an authentication service object, which permits providers to digitally sign communications objects and permits consumers to automatically verify these digital signatures. An example of the third category is a payment service object, which permits a provider to automate receiving payments from a bank or credit company and permits consumers to automate sending payments to the bank or credit company. Alternatively, where it is more efficient, service objects can be split into provider/consumer "pairs", each containing a link component object 110 linking it to the other.

Detailed Description Text (335):
Service objects can wholly contain the services they offer, or they can represent the services of one or more servers available in the communications object system. In the latter case, the service object forms the basis for communicating the necessary information to the server so that the service can be provided. The latter case is also particularly useful because the service object can abstract or "encapsulate" the interface to the server or servers, removing the need for either the provider or consumer to deal with this complexity. A service object may represent services provided by a network of related servers, for example a replicated directory database such as the Internet's Domain Name Service (DNS). The service object can then also abstract and automate the process of choosing one of the network servers as a current partner server which will result in optimal performance and minimal network traffic. Referring to FIG. 3, this feature is accomplished by including a receipt method 141 in the service object (815, FIG. 17) having one or more algorithms for determining the optimal partner server. Such algorithms can access elements 143 in the provider database 11 or consumer database 21 for necessary input data such as the user's geographic location, time zone, network address, and so on, or they can prompt the user for such data via input forms. The service object's receipt method may also use a data exchange method 141 to query a reference server, perform network packet timing tests, or use other techniques to determine the optimal partner server. The same approach can also be used to determine one or more backup partner servers to use in case the primary partner server is unavailable. The particular algorithm or method for determing an optimal partner server or backup partner servers is not a limiting feature of the invention. Once determined, the receipt method can save this data in the provider database 11 or consumer database 21 as one or more element preference instances 147. These element preference instances can then be accessed by the service object's update method 141 or any of its other methods whenever it needs to call

operations at the partner server.

Detailed Description Text (336):
Service objects are distinguished in a communications object system by their communications object type. Examples of service object types are shown as classes 830-844 in FIG. 17. Service object types serve the same function at the communications object level as element types serve at the element level. Service object typing allows the programs 12, 22 and other communications objects to make calls to service objects by type. When more than one service object of the proper type is present, the programs 12, 22 can prompt the provider or consumer to choose the desired service object, or the programs 12, 22 can make the choice programmatically. For example, a provider seeking to list a new communications object in one or more directories could execute a system method 141 that would call all directory service objects 832 present in the provider database 11. If more than one was present, the system method 141 would generate an input form prompting the provider to choose the desired directory service object or objects 832. When this form was submitted by the provider, the system method would proceed to call a DirectoryList method of each directory service object 832. The DirectoryList method for each directory service object would then carry out the balance of the operations needed to complete the directory listing.

Detailed Description Text (337):
The same fundamental data structures used in the provider database 11 and consumer database 21 can be used in a partner server database 1301. These data structures are shown in FIGS. 3 and 4. Service objects (815, FIG. 17) can be stored as communications objects 110. They can also be nested as composite and component objects (811, 812, FIG. 17) using association 110A as explained above in the description of composite and component objects. The use of composite and component service objects is ideally suited to many partner server functions, as further explained below. Alternatively, partner servers can use other data structures in order to optimize database performance, particularly for high-volume applications. Such additional data structures may include the use of special indexes or indexing algorithms, special caches or cache techniques, and other database performance enhancement technologies.

Detailed Description Text (339):
As a subclass of standard communications objects 110, service objects can include all the control functions of communications objects described above. Certain control functions have special relevance for service objects. First, link control allows other communications objects to call the methods of a service object object regardless of where the service object may be located on a communications network 3. The special applications of link control will be discussed below. Second, update control allows a service object to stay current regardless of where it is located on a communications network 3. Version monitoring and update querying are particularly efficient techniques of update control for service objects and will be discussed below. Third, notification control allows a service object provider to notify providers or consumers using the service object about relevant changes to the service object or the communications services it makes available. Fourth, data exchange control allows the service object to automate data exchanges with the server or servers the service object may represent. Fifth, data archive control allows service objects to delete themselves if they age beyond a certain date or have not been used within a certain period. This allows databases 100 to avoid an accumulation of seldom-used service objects. Finally, event tracking control and reporting control allows service objects to create and report transaction records which can be processed to provide further services to the provider or consumer. These transaction records can also be used by the service object provider for billing or statistical purposes.

Detailed Description Text (340):
Link control and update control have special applications to polymorphic service

objects. The application of link control to polymorphic service objects is illustrated in FIG. 28. A provider using the provider program 12 has need of the services offered by a service object partner server 1302. First the provider obtains a service object 1310 from the partner server 1302 (step 1320). This may be by browsing with a web browser 50, receiving the service object 1310 via e-mail, or any of the other techniques described above. Such partner server 1302 may be a distribution server 32 or any other type of service object partner server such as those described below. Once the provider has obtained the service object 1310, the provider may add a link component object 110 to any of the provider's communications objects 110 which need to access the elements or methods the service object 1310. This link component object 110 will then be included in any communications object instance 35 generated from the consumer database 21 (step 1321). In a preferred embodiment, the link component object 110 is supplied by the service object 1310 itself. In this case, referring to FIG. 3, the provider need only create a contained-by association between the link component object 110 in the service object (1310, FIG. 28) and the provider's communications object 110. This association can also be created automatically when any service object method 141 is executed that creates a service relationship between the service object and the communications object 110. The communications object 110 thus becomes a synthesized object (813, FIG. 17), wherein the link component object 110 is supplied by the service object 1310. Examples of such a service object relationship include listing a communications object 110 in a directory server, registering a communications object 110 with an authentication server, or authorizing a communications object 110 for use with a payment server. Further examples will be given below. Referring again to FIG. 28, the next step is that a communications object instance 35 is transferred to a consumer program 22 (step 1322). This may be via e-mail using the push technique, via a distribution server 32 using the pull technique, or any of the other techniques described above. Once the communications object instance 35 is transferred to the consumer program 22, a link method 141 of the link component object 110 may be manually executed by the consumer or automatically executed by another system method or communications object method. For example, the consumer may wish to look up related communications objects instances 35 in a directory server, or authenticate the communications object instance 35 before forwarding it, or make a payment transaction using the communications object instance 35. When the link method 141 is executed, it uses the attributes of the link element 143 to locate the designated service object 1310 as described in the communications object exchange control section above. For example, if the service object 1310 is not present locally in the consumer database 21, the link method uses other attributes of the link element to locate the service object 1310. For instance, if a URL was present, the link method would use it to obtain the service object 1310. If this fails, the link method would use the UID or name of the service object 1310 to obtain its URL or other current network address via a name server. The link method could also call the methods of a name service object, described below. Once the link method located the proper network address, it would download the service object 1310 from the partner server 1302 (step 1323). At this point the link is reestablished, and the communications object instance 35 can call the service methods of the service object 1310 to perform the services requested (step 1324).

Detailed Description Text (341):
Once a service object 1310 has been transferred to a consumer database 21 via this technique, version monitoring can be a very efficient update control technique. This is because the services of the service object 1310 may not be required again until they are called by another communications object instance 35. Version monitoring can be employed as follows. First, the service object 1310 can use a standard push or pull update control technique at the provider program 12 to maintain a current version. Such version changes will also maintain current version values in any link element 143 associated with the service object 1310. By including this link element 143 in any update of a communications object 110 which contains it, the link element 143 will be transferred to all recipient consumer programs 22 when the communications object 110 is updated. At this point, whenever

a method call is made from a communications object instance 35 to the service object 1310, a version monitoring rule 140 contained in the service object 1310 can be triggered. The version monitoring rule 140 compares the service object version value stored in the link element of the calling communications object 110 with the version value of the service object 1310. If the version value in the link element 143 is greater than the version value of the service object 1310, the update method of the service object 1310 is executed and the service object 1310 is updated prior to completion of the original service object method call.

Detailed Description Text (342):
Update queries are also a highly efficient update control technique for service objects. This is especially true when a service object 1310 is used to represent access to a large database of communications objects 110, such as the categories of a yellow pages directory server or a classified advertising server. Basic update query control is explained in the update control section above. When used in conjunction with a service object 1310 and partner server 1302, update query control can be even more efficient by employing user objects 110 on the partner server 1302. The data structures for user objects are shown in FIG. 6B. User objects 110 represent either the providers or consumers interacting with the partner server 1302. The partner server 1302 maintains an index of the provider and consumer relationship associations 111 between all communications objects 110 and the user objects 110. Updates to communications objects 110 in the partner server database 1301 set the NewFlag attribute value of the relationship association to TRUE. This can be accomplished very efficiently on the partner server 1302 using an update association rule 140 and the update association routine (FIG. 10B). By employing such a user object index, an update query method 141 of a service object 1310 need only submit the provider UID in its update query. The partner server 1302 executes the query against the user object index to determine all communications objects 110 associated with that user UID where the NewFlag attribute value is TRUE. Those communications objects 110 are returned immediately as the query result set and the NewFlag attribute for each of these relationship associations 111 is reset to FALSE. User objects 110 can also act as recipients 120. In this case the partner server 1302 can transmit communications object updates via the push technique. The data and methods in user objects 110 on a partner server 1302 can be automatically kept current by a data exchange rule 140 in the service object 1310 as explained in the data exchange control section. The use of user objects 110 in a communications object system can be better understood in the discussion of multiuser operation in the advanced architecture sections below.

Detailed Description Text (343):
The following sections will explain how service objects and partner servers can be employed to provide registration, maintenance, name, directory, distribution, encoding, authentication, data exchange, payment, reporting, and feedback services for a communications object system. This set of service object applications is not exhaustive but merely illustrative of how service objects may be employed. Service objects and partner servers may also combine any number of services. Alternatively, any of the services described below may also be performed directly by one or more system methods 141, rules 140, and elements 143 instead of service objects 1310. Service objects are a preferred embodiment because they allow such services to be encapsulated, distributed, optimized, and updated throughout a communications network 3.

Detailed Description Text (345):
A registration service object type (830, FIG. 17) can be used to obtain the initial database system ID (100, FIG. 3) used to uniquely identify a provider database 11 or consumer database 21. This process is explained in the system ID and naming services section above. As illustrated in FIG. 5, if a communications object system only requires one system ID server 42 (also called a registration partner server), registration services are easily be accomplished using a system method 141. However, this same process can also be used to obtain other system IDs which can

function as group IDs, registration keys, licensing keys, and so on. In this case multiple registration partner servers may be desirable. For example, in addition to a global Internet-wide registration server, a company may wish to have its own registration partner server to manage communications object system group IDs and authorization keys for its employees. By creating its own registration service object, the company can quickly and easily add these services to the programs 12, 22.

Detailed Description Text (346):
Registration partner servers can do more than just automate and track system ID and group ID assignments. By including data exchange methods for other registration data, such as names, addresses, contact information, and so on, registration partner servers can obtain and store all the information necessary to fully register communications object system users. By using data exchange methods to return registration keys and licensing keys to the databases 11, 21 which enable the operation of all or selected subsets of features, registration partner servers also function as licensing servers. For this reason registration servers can be efficiently coupled with naming, authentication, and reporting service objects and partner servers.

Detailed Description Text (350):
Maintenance service objects 1310 also allows all the providers in a communications object system to contribute and obtain new system components. This is accomplished by including a data exchange method 141 in the maintenance service object 1310 that automates the process of uploading and registering the new components in a database 1301 at a maintenance partner server 1302. Another data query method 141 in the same maintenance service object 1310 allows other providers to manually or automatically search the maintenance partner server database 1301. This process is fully described in the data exchange service object section below. The sharing of communications object components across a communications object system is particularly valuable in relation to rules 140, methods 141, and type definitions 144. New rules and methods allow providers to extend the functionality of the communications object system easily. For example, new system objects can encapsulate the services of particular communications protocols. These include network protocols, such as TCP/IP, IPX/SPX, and NetBEUI; communications protocols, such as XMODEM, YMODEM, HTTP, NNTP, and SMTP; APIs such as TAPI, MAPI, and Visual Basic; and even device driver protocols such as those required for printers, modems, CD-ROM drives, monitors, and so on. The particular protocols used are not a limiting feature of the invention. Since newer, more powerful, more efficient, and more secure protocols are always evolving, encapsulating these in component communications objects that can be easily distributed throughout a communications object system is a major advantage.

Detailed Description Text (351):
New type definitions allow groups of providers to create shared data structures that meet particular needs, such as specialized electronic data interchange (EDI) standards for vertical markets. Shared type definition repositories have particular applications that enable new types of intelligent information interchange. An example discussed above in the data exchange control section is the customization of web server content for web browser users. This interchange can be substantially enhanced through the establishment on a maintenance partner server 1302 or a distributed network of partner servers 1302 of a server database 1301 of shared type definition instances 144. These type definition instances could cover families of common psychographic variables such as political affiliations, color preferences, food and beverage preferences, entertainment preferences, and so on. Using a maintenance service object 1310, providers can search for and download these type definitions 144 to be incorporated into the design of the provider's web content customization system as well as the provider's own communications objects 110. When a consumer browses the provider's web server, the web server can return a communications object 110 that queries the consumer database 21 for the values of

elements 143 corresponding to specified psychgraphic type definitions 144. If these type definitions are not present, the communications object 110 will include a link component object 110 to obtain the necessary type definitions 144 from the maintenance server 1302. The communications object 110 then generates an input form requesting the values for elements 143 corresponding to each type definition. Once these element preference instances 147 are saved in the consumer database 21, the consumer need not enter this pschographic preference data again. It will be available automatically to any other provider who needs it, subject to data access rules 140 applied by the consumer. In this way the consumer database 21 can steadily grow "smarter" about the consumer's interests and tastes, and providers have a highly direct, efficient, and automated mechanism with which to obtain such psychographic data from the consumer.

Detailed Description Text (355):
The use of naming and name resolution services in a communications object system are fully discussed in the communications object exchange control section above. A name partner server 1302 can implement name services using many database structures. In a preferred embodiment, name services for individuals are provided using communications objects 110 of the user object type (816, FIG. 17). Referring to FIG. 3, a basic "white pages" name service can be implemented using an element 143 of a composite type UniqueName for each user object 110 listed. A user object 110 could have more than one unique name by allowing more than UniqueName element 143 to be associated with the user object 110. Alternate names may also be desirable, for example to allow individuals using nicknames to be located using the nickname. Alternative names are also advantageous in a commercial tradename name service because it allows companies who use the same trademark name across different industries to all reference that name. Alternate names can be created using an element 143 of the communications object 110 with the composite type AlternateName. By including methods 141 for listing and editing a name as well as searching for a unique name or an alternate names, a name service object 110 can automate name server access for both providers and consumers.

Detailed Description Text (358):
Directory Service Objects and Partner Servers

Detailed Description Text (359):
A directory service object type (833, FIG. 17) and a directory partner server 1302 provide an extension to name services whereby communications objects can be listed and located by additional attributes. Any attribute that enables consumers to locate communications objects of interest may be employed. For example, consumers may wish to locate user objects 110 representing other consumers. In this case, desirable attributes may include geographic location, age, occupation, family lineage, educational affiliation, political affiliation, religious affiliation, and so on. These attributes may all be represented by different elements 143 in a directory partner server 1302 in the same manner as described above for psychographic attributes on a type definition maintenance server. When a provider of a communications object 110 uses a directory service object 1310 to create a directory listing on the directory partner server 1302, a data exchange method 141 in the service object uses the system ID of the type definition 144 for each required or desired attribute to automatically identify and copy these attributes from elements 143 in the provider database 11 to elements 143 in the directory partner server database 1301.

Detailed Description Text (360):
Another commonly desired set of directory attributes is a hierarchical categorization system such as that employed by many "yellow pages" directories. In a preferred embodiment, such a categorization system is implemented on a directory partner server 1302 using composite communications objects (811, FIG. 17) and component communications objects (812, FIG. 17). Such a data structure is illustrated in FIG. 29A. The directory service object 1401 functions as the

highest-level composite communications object. Each of its first-level component objects represents the top-level categories 1410, 1411, 1413. These component objects are called category objects. Component objects of each top-level category object represent the second-level categories 1421, 1422, 1423, 1424. Such a category object structure can be nested as many layers deep as is necessary. Alternatively, a separate directory service object 1401 could represent a particular branch within the category structure.

Detailed Description Text (361):
The use of a communications object-based directory service offers several advantages over conventional directory systems. First, it can simplify and automate the listing process for providers. The steps in this process are illustrated in FIG. 30. In this example a directory system is implemented on a directory partner server 1302 as a web server using HTML pages to display the category hierarchy. Each category description includes a hyperlink to its category object 110. The process begins with the provider using his/her browser 50 to navigate the directory partner server 1302. The provider chooses the hyperlink representing each category object 110 in which the provider is interested in listing a communications object 110 (step 4001). The receipt method for the category object 110 first checks to see if its parent directory service object 1310 is present in the provider database 11 (step 4002). If not, the category object uses its link component object 110 to download the directory service object 1310 from the directory partner server 1302 (step 4003). Next the receipt method 141 for each category object 110 generates an input form prompting the provider for the communications object or objects 110 to be listed in this category (step 4004). When this input form is submitted, the receipt method creates an association between the communications object or objects 110 and the category object 110 (step 4005). Finally, the receipt method 141 asks the provider if he/she would like to choose additional category objects (step 4006). If so, the above steps are repeated. Once the provider has chosen all desired category objects 110, the provider executes a data exchange method 141 in the directory service object 1310 that will carry out the listing procedure (step 4007). This data exchange method 141 queries the provider database 11 for all category objects 110 belonging to the directory service object 1310 (step 4008). The data exchange method 141 then queries for all of the provider's communications objects 110 associated with these category objects 110 (step 4009). The data exchange method 141 creates a message object 110 containing this query result set together with the necessary attributes or elements of the listed communications object or objects 110 (step 4010). The data exchange method 141 transmits this message object 110 to the directory partner server 1302 (step 4011). When received by the directory partner server 1302, the message object 110 triggers a corresponding data exchange method 141 (step 4012). This data exchange method 141 creates or modifies the listing for each communications object or objects 110 in the directory partner server database 1301 (step 4013). This listing consists of a new component object 110 containing the desired listing elements 143. The data exchange method 141 then creates the appropriate associations with each composite category object 110 (step 4014). When this process is completed, the data exchange method 141 creates a message object 110 containing an appropriate acknowledgment message (step 4015). The data exchange method 141 transmits this message object 110 back to the directory service object 1310 at the provider program 12 (step 4016). When received by the directory service object 1310, the message object 110 executes its receipt method 141 (step 4017). The receipt method 141 executes the provider's desired notification method 141 to complete the listing (step 4018). This directory listing process can also include authentication services, payment, or reporting services as further described below.

Detailed Description Text (362):
A second advantage of a communications object-based directory service is that it automates the directory updating process in both directions. The steps in the process of updating a provider's listings on a directory partner server 1302 are shown in FIG. 31A. This process employs a data exchange rule 140 in the directory

service object 1310, explained in the data exchange control section above. The data exchange rule 140 monitors for changes in attributes or elements 143 of a communications object 110 associated with a category object 110 or a directory service object 1310 (step 4031). When such changes occur, the data exchange rule triggers a data exchange method 141 in the directory service object 1310 (step 4032). The data exchange method 141 creates a message object 110 containing those changes (step 4033). The data exchange method 141 transmits the message object 110 to the directory partner server 1302 (step 4034). When received by the directory partner server 1302, the message object 110 triggers a corresponding data exchange method 141 (step 4035). This data exchange method 141 updates the necessary attributes and associations in the directory partner server database 1301 (step 4036). If desired, data exchange method 141 also returns an acknowledgment message object 110 to the directory service object 1310 at the provider program 12 to confirm the update has been made.

Detailed Description Text (363):
The steps in the process of notifying a provider about changes to one or more category objects on the directory partner server 1302 are shown in FIG. 31B. Such changes occur when a category definition is changed, the directory provider needs to bifurcate a directory category into two or more categories, when a category is replaced by another category or categories, and so on. This process uses the update query technique described in the update control section above to monitor the directory partner server 1302 for changes. When the first directory listing is made by the directory service object 1310, the data exchange method 141 creates a scheduled event instance 117 in the provider database 11 (step 4051). When activated, the scheduled event instance 117 triggers a update query method 141 in the directory service object 1310 (step 4052). The update query method 141 first queries the provider database 11 for the UID and version value of all category objects 110 associated with the directory service object 1310 (step 4053). Alternatively, an index of these values could be maintained as an element in the directory service object 1310. The update query method 141 then creates a message object 110 containing the result set (step 4054). The update query method 141 transmits this message object 110 to the directory partner server 1302 (step 4055). When received by the directory partner server 1302, the message object 110 triggers a corresponding update query method 141 (step 4056). This update query method 141 uses the message object result set to query the directory partner server database 1301 for any changes to the corresponding category objects 110 (step 4057). The update query method 141 then returns the result set to the provider program 12 (step 4058). If there are no changes, the result set is a message object 110. If there have been changes, the result set is the changed category objects 110. The provider program 12 receives the result set and executes any receipt methods pertaining to the result set objects (step 4059). This includes the notification test (step 4060). If the provider desires notification of changes to directory category objects upon which the provider may wish to take action, the provider program 12 executes the desired notification methods (step 4061).

Detailed Description Text (364):
Alternatively, for high-volume applications, the directory partner server 1302 can maintain an index of the provider UIDs associated with each category object 110. These UIDs can be stored in user objects 110. In this case step 4053 can be eliminated, and the update query in step 4054 can consist of just the provider UID. The provider user objects 110 can also function as recipients 120 on the directory partner server 1302. In this case, updated category objects can be distributed using the push technique. This process is explained in the service object introduction section above.

Detailed Description Text (365):
A third advantage of a communications object-based directory service is that consumers may use a directory service object 1310 to monitor a directory partner server 1302 for new listings in any category or changes to the category structure.

Because they are symmetric and can be performed by a polymorphic directory service object 1310, these processes are largely identical to those described above for a provider. In addition, data exchange methods in a directory service object 1310 can allow consumers to create custom queries that can be run at scheduled intervals against the directory server 1302. Thus a consumer could, for example, be notified if any new listing containing the word "Mustang" was added to a directory server 1302 even if there was no "Mustang" category object 110. This is further discussed in the section on data exchange service objects below.

Detailed Description Text (366):
The value of a communications object-based directory services can be further increased using link control. Any provider who associates (lists) a communications object 110 with one or more category objects 110 on a directory partner server 1302 can include a link component object 110 from those category objects 110 in the communications object 110. This is identical to the process of including a link component object 110 from a service object 1310 as described above. Category object links provide a powerful new way for consumers to locate communications objects in which they are interested. The consumer can just activate the link method 141 to immediately download the desired category object 110 and access directory listings for other communications objects 110 in the same category. Using the directory service object 1310 linked to the category object 110, the consumer can also immediately begin monitoring the directory partner server 1302 for new listings in that category.

Detailed Description Text (367):
Directory partner servers are well-suited to be combined with distribution partner servers and data exchange partner servers because it is easy to create and maintain associations in a single database 100 between directory category objects 110, the listed communications objects 110, elements 143 associated with the communications objects 110, and user objects 110.

Detailed Description Text (368):
Since a communications object can represent anything which a provider wishes to communicate, the advantages of a communications object system directory service can be transfered to any real-world function where directory services are useful. Besides conventional white pages and yellow pages, this includes catalogs, professional and academic directories, computer network directories, personal address books, classified advertising services, and so on. The specific directory service is not a limiting feature of the invention.

Detailed Description Text (370):
A distribution service object type (834, FIG. 17) can automate the transmission, storage, and updating of communications objects on a distribution partner server 1302. While any server on a communications network 3 capable of file storage and retrieval can operate as a distribution server 32, a distribution partner server 1302 which includes the full capabilities of a database 100 offers many additional services to communications object providers and consumers.

Detailed Description Text (372):
The inverse of this process becomes a key advantage for consumers because a single distribution service object 1310 can use the update query technique to monitor a distribution partner server 1302 for updates to all communications objects 110 associated with the distribution service object 1310. The greater the number of communications objects 110 associated with a single distribution service object 1310, the more efficient the update process becomes. Only a single update query needs to be made to the distribution partner server 1302. This update query technique is further described in the update control and directory service object sections above. This technique is particularly efficient when used in conjunction with a user object 110 index at the distribution partner server 1302. User object indexes are explained in the service object introduction section above.

Detailed Description Text (385):
An example is authentication using digital signatures based on public/private keys. The first set of steps in this process are shown in FIG. 32A. The process begins with the provider obtaining a suitable authentication service object (1310, FIG. 28) if one is not already present in the provider program 12 (step 4101). An authentication service object 1310 contains one or more public keys from its corresponding authentication partner server 1302, stored as elements 143. The authentication service object 1310 also contains the encoding method or methods 141 necessary to carry out its authentication functions, called authentication methods. When the provider is ready to create an authentication account, the provider executes one of the authentication methods 141 to generate a public/private key pair (step 4102). The private key is stored as an element 143 of the authentication service object 1310 in the provider database 11 (step 4103). Optionally, the data exchange method 141 may also encrypt this private key element 143 with a password known only to the provider and not stored anywhere in the provider database 11 or on the local computer. The authentication method 141 next creates an authentication order consisting of three elements: the public key generated in step 1402, the provider's UID, and a unique registration key known only to the provider and the authentication partner server 1302 (step 4104). Other elements or variables, such as a timestamp, may also be included. If the authentication partner server 1302 is operated in conjunction with a registration partner server 1302, the unique registration key may be the provider's password or other identification key created at the time of registration. This is shown as the Key attribute of the system ID instance (251, FIG. 6A). This unique registration key is stored in the provider database 11 as an encrypted element 143 which can be decrypted using a provider-supplied password. Alternatively, it may not be stored at all locally but be entered manually by the provider when required. The authentication method 141 next encrypts the authentication order using the authentication partner server's public key (step 4105). The authentication method 141 then creates a message object 110 containing the encypted authentication order (step 4106). The authentication method 141 transmits this message object 110 to the authentication partner server 1302 (step 4107). The authentication partner server 1302 receives the message object 110 and executes its receipt method 141, which is either the same authentication method or another authentication method residing on the authentication partner server 1302 (step 4108). This authentication method 141 decrypts the authentication order using the authentication partner server's private key (step 4109). Next the authentication method 141 verifies the provider's unique registration key and UID in the authentication partner server database 1301 to validate the authentication order (step 4110). The authentication method 141 then creates a public key certificate by combining the provider's public key with certain other identifying data, such as the provider's UID (step 4111). The authentication method 141 digitally signs the public key certificate using the authentication partner server's private key (step 4112). The authentication method 141 then creates a message object 110 containing the public key certificate (step 4113). Finally, the authentication method 141 transmits the message object 110 back to the authentication service object 1310 in the provider program 12 (step 4114). There the provider program 12 receives the message object 110 and executes the original authentication method 141 in the authentication service object 1310 (step 4115). This authentication method 141 first verifies the signature of the public key certificate using the public key of the authentication partner server 1302 (step 4116). Lastly the authentication method 141 saves the public key certificate in the provider database 11 as an element 143 (step 4117).

Detailed Description Text (389):
Authentication on a communications object system may also take place without using centralized authentication partner servers 1302. This technique, known as distributed key management, is used by the public-domain encryption program Pretty Good Privacy (PGP). It is based on the concept of an "introducer". An introducer is a person who signs the public key certificate of another person whose identity they

personally know and are willing to certify. Introducers are easily employed on a communications object system using authentication service objects 1310. The steps in the process for using introducers are illustrated in FIG. 33A. First, a user requring a public key certificate introduction, called the "originator", executes a data exchange method 141 of an authentication service object 1310 to generate a public/private key pair (step 4151). Next, the data exchange method 141 stores each key as an element 143 of the authentication service object 1310 (step 4152). Then the data exchange method 141 creates a public key certificate consisting of the public key element 143 plus such additional elements 143 as will allow any potential introducer to certify the identify of the orginator (step 4153). These first three steps can be omitted if the originator only wishes to add introducers for an existing public key certificate already stored as an element 143 of the authentication service object 1310. Now, the data exchange method 141 generates an input form prompting the originator for the recipients 120 whom the originator would like to make introduction requests (step 4154). The checkboxes on this input form can represent each of the recipients 120 in the originator's consumer database 21, or the originator can specify the e-mail addresses of still other potential introducers. The input form also allows the originator to enter the attributes of a message element (211, FIG. 4) to be sent to these recipients. When the input form is submitted, the data exchange method 141 creates a message object 110 consisting of the public key certificate, the message element, and any other relevant data, such as a timestamp (step 4155). The data exchange method 141 transmits this to all recipients 120 selected by the originator (step 4156). When received by the recipient's consumer program 22, the message object's receipt method 141 executes the recipient's selected notification method or methods 141 for introduction requests (step 4157). If distributed key management was implemented on a communications object system, message objects containing introduction requests can use a standard notification element type definition 144. This type definition 144 allows consumers to assign notification methods 141 globally for all introduction requests, or designate specific notification methods for introduction requests from individual recipients 120. When the recipient responds to the notification message, a data exchange method 141 in the authentication service object 1310 is executed (step 4158). This data exchange method 141 generates a input form for confirming the introduction request from the originator (step 4159). This input form may include any such data as may be relevant to an introduction request, including the elements 143 of the public key certificate that fully identify the originator. The recipient may also wish to verify the public key with the originator via another secure channel, such as via telephone. When the recipient is satisifed that the request is genuine, the recipient submits the input form (step 4160). The data exchange method 141 calls an authentication method 141 in the authentication service object 1310 which digitially signs the originator's public key certificate using the recipient's private key (step 4161). If the recipient's private key is stored as an encrypted element 143 of the authentication service object 1310, the recipient may need to enter password or passphrase for decryption. Then the data exchange method 141 creates a message object 110 containing the signed public key certificate (step 4162). The data exchange method 141 transmits this message object 110 to the originating authentication service object 1310 at the originating consumer program 22 (step 4163). When the message object 110 is received, the consumer program 22 executes the originating data exchange method 141 (step 4164). This data exchange method 141 stores the signed public key certificate as an element 143 of the authentication service object 1310 (step 4165). Finally, the data exchange method 141 executes any notification methods 141 assigned by the originator to the acknowledgment of introduction requests (step 4166).

Detailed Description Text (390):
Once a set of signed public key certificates has been received by the originator, the originator can send a public key acceptance request to any other communications object system user. The steps in the process for public key certificate acceptance requests are illustrated in FIG. 33B. The originator initiates the request by executing a data exchange method 141 of an authentication service object 1310 (step

4181). This data exchange method 141 generates an input form for the acceptance request (step 4182). This input form can include the attributes of a message element (211, FIG. 4) allowing the originator to compose the electronic equivalent of an introductory letter. The input form can also allow the originator to choose the introducers whose public key certificate signatures the originator wishes to present to the recipient. When the input form is submitted, the data exchange method 141 creates a message object 110 consisting of the selected public key certificate signatures, the message element, and any other relevant data, such as a timestamp (step 4183). Note that the first two steps above may be omitted if the acceptance request comes directly from another communications object method 141. In this case the recipient of the acceptance request will be specified in the method call, the set of introducer signatures can be selected algorithmically, and the message object in step 4183 can be created automatically. Next the message object 110 is transmitted to the recipient 120 (step 4184). When received by the recipient's consumer program 22, the message object's receipt method 141 executes a data exchange method 141 of an authentication service object 1310 (step 4185). This data exchange method 141 compares the UID of the introducer public key certificate signatures in the message object 110 with the UID of the trusted public key certificates stored in the recipient's consumer database 21 (step 4186). These trusted public key certificates are stored as elements 143 of the authentication service object 1310, and represent introducers whom the recipient trusts. For any matching UIDs, the data exchange method 141 then calls an authentication method 141 to verify the introducer signature using the introducer's public key (step 4187). The data exchange method 141 then checks an acceptance request preference element 147 in the recipient's consumer database 21 to determine if notification is desired (step 4188). For example, notification may not be desired if the signatures of 3 or more introducers are verified. If notification is desired, the data exchange method 141 executes the assigned notification methods 141 to generate a notification message for the recipient (step 4189). When the recipient responds to the notification message, a data exchange method 141 in the authentication service object 1310 is executed (step 4190). This data exchange method 141 generates a input form for confirming the acceptance request from the originator (step 4191). This input form can include the results of the comparison test in step 4186. It can also include input fields for a message back to the originator, messages to the introducers, or other automated options. For purposes of this illustration, we will assume the recipient confirms the acceptance request when the input form is submitted (step 4192). (If the recipient denies the request, the following steps could produce a negative acknowledgment message to the originator.) The data exchange method 141 then saves the originator's public key certificate as an element 143 of the authentication service object 1310 (step 4193). This now becomes another of the recipients trusted public key certificates. The data exchange method 141 next creates a message object 110 containing an acknowledgment of the acceptance request (step 4194). Optionally, this message object 110 could also include a copy of the originator's public key certificate signed by the recipient using the recipient's private key. The data exchange method 141 transmits this message object 110 to the originating authentication service object 1310 at the originating consumer program 22 (step 4195). When the message object 110 is received, the consumer program 22 executes the originating data exchange method 141 (step 4196). This data exchange method 141 stores the public key certificate acceptance acknowledgment as an element 143 of the authentication service object 1310 (step 4197). Such acceptance acknowledgments can now be checked automatically by data exchange methods 141 in the consumer program 22. Alternatively, if the acceptance acknowledgment included a copy of the originator's public key certificate signed by the recipient, this public key certificate could be added to the originator's set of introducers. Finally the data exchange method 141 executes any notification methods 141 assigned by the originator to the acknowledgment of acceptance requests (step 4198).

Detailed Description Text (396):
An example of the first case is a product registration service object 1310. A

software company may offer a large number of software products, each with its own representative communications object 110. If the software company wanted all these communications objects to share the same product registration services for new buyers of the company's software products, the company could create a product registration service object 1310. Any of the company's product communications objects 110 could then call the services of the product registration service object 1310 to carry out a new user product registration via a product registration partner server 1302. Ideally, such a shared service object would also offer other common services to the individual product objects, such as distribution/update control, directory services, and so on.

Detailed Description Text (397):
For a communications object system deployed on a wide area network, such as the Internet, there are a number of common data exchange services desired by many providers. Besides the specialized services discussed above, examples include file transfer, fax transfer, news distribution, discussion databases, knowledgebases, and classified advertising services. In most cases polymorphic service objects 1310 are desirable for data exchange. This is because the same data exchange service objects 1310 that allow communications object system users to upload and maintain data at a data exchange partner server 1302 can allow other communications object system users to automatically monitor and/or download that data as desired. A simple example is an FTP service object 1310 and an FTP partner server 1302 offered by a provider of network file backup services. The FTP service object 1310 would allow users to select a local file or files which the FTP service object 1310 would monitor and automatically transfer to the FTP partner server 1302 at periodic intervals or when the files had changed. The same FTP service object 1310 could be used to restore backed up files from the FTP partner server 1302 to the user's local system. The FTP service object 1310 could combine these backup services with payment and reporting services. Payment and reporting services are discussed below.

Detailed Description Text (398):
A more advanced example is classified advertising services. A classified ad service object 1310 combines the functions of a data exchange service object 1310 and a directory service object 1310. (It could also incorporate the functions of an authentication service object 1310, payment service object 1310, reporting service object 1310, or other such service object functions as may be applicable.) A classified ad partner server 1302 represents the categories of the classified advertising system as category objects in the same manner as a directory partner server 1302 (FIG. 29A). Each of these category objects 110 includes one or more elements 143, methods 141, and rules 140 that allows a classified advertiser to define the attributes and values of an ad listing in this category.

Detailed Description Text (399):
The steps involved in the process of a seller using a classified ad service object 1310 to create an ad listing in a classified ad partner server database 1301 are shown in FIG. 34A. (These closely parallel the steps involved in creating a listing on a directory partner server 1302 shown in FIG. 30.) The process begins with the seller using his/her browser 50 to navigate the classified ad partner server 1302. The seller chooses the hyperlink representing the category object 110 in which the provider is interested in making an ad listing (step 4201). The receipt method for the category object 110 first checks to see if its parent classified ad service object 1310 is present in the provider database 11 (step 4202). If not, the category object uses its link component object 110 to download the classified ad service object 1310 from the directory partner server 1302 (step 4203). The receipt method 141 then executes a data exchange method 141 in the classified ad service object 1310 that generates a listing input form (step 4204). This input form consists of the category attribute and value choices obtained from the category object 110. For example, a category object 110 such as "Minivan" might generate an input form for attributes including make, model, year, color, mileage, condition,

and so on. The appropriate value choices for each of these attributes would be displayed as drop-down lists, radio buttons, and so on. When the seller submits the completed input form, the data exchange method 141 creates a message object 110 containing this input data (step 4205). This message object 110 also contains such other communications object components as are necessary to let prospective buyers communicate with the seller. The data exchange method 141 then transmits this message object 110 to the classified ad partner server 1302 (step 4206). When received by the classified ad partner server 1302, the message object 110 triggers a corresponding data exchange method 141 (step 4207). This data exchange method 141 first creates a communications object 110 representing the seller in the classified ad partner server database 1301 (step 4208). Next the data exchange method 141 creates a communications object 110 of the component object type (812, FIG. 17) containing the ad listing element or elements 143 (step 4209). Then the data exchange method 141 creates member associations 110A between the seller's composite communications object 110 and the category composite communications object 110 (step 4210). In this manner the seller can submit additional ad listings without needing to duplicate the data in the communications object 110 representing the seller. When this process is completed, the data exchange method 141 creates a message object 110 containing an appropriate acknowledgment message (step 4211). The data exchange method 141 transmits this message object 110 back to the classified ad service object 1310 at the provider program 12 (step 4212). When received by the classified ad service object 1310, the message object 110 executes its receipt method 141 (step 4213). The receipt method 141 executes the seller's desired notification method 141 to complete the ad listing process (step 4214). This process could also incorporate authentication services as described above or payment and reporting services as described below.

Detailed Description Text (400):
Any interested buyer can use the same classified ad service object 1310 and category object 110 to specify and monitor ad listings that meet the buyer's interests. The steps involved in this process are shown in FIG. 34B. (This process is similar to the process of monitoring category objects 110 on a directory partner server 1302 as shown in FIG. 31B.) As with the ad listing process, the monitoring process begins with the buyer using his/her browser 50 to navigate the classified ad partner server 1302. The buyer chooses the hyperlink representing the category object 110 in which the buyer is interested in making a purchase (step 4231). The receipt method for the category object 110 first checks to see if its parent classified ad service object 1310 is present in the consumer database 11 (step 4232). If not, the category object uses its link component object 110 to download the classified ad service object 1310 from the directory partner server 1302 (step 4233). The receipt method 141 then executes a data exchange method 141 in the classified ad service object 1310 that generates a monitoring input form (step 4234). The monitoring input form is largely identical to the listing input form described above. It draws some of its attributes and values from the category object 110. The principle difference is that it allows the buyer to specify value ranges or other query formulas for category attributes obtained from the category object 110. To use the automobile example above, a "Minivan" category object might use drop-down list of integer values for the "Not older than" year attribute; use checkboxes for multiple color choices; accept an integer value for "maximum mileage"; use radio buttons for acceptible condition attributes; and so on. When the form is submitted, the data exchange method 141 first saves the input form data as a query element 143 (step 4235). Secondly it creates one or more scheduled event instances 117 in the consumer database 11 (step 4236). These scheduled event instances 117 can begin immediately and repeat at intervals or according to rules 140 specified by the consumer on the input form. They can also be subject to monitoring rules 140 imposed by the classified ad service provider in the category object 110 or classified ad service object 1310. When activated, these scheduled event instances execute a data exchange method 141 in the classified ad service object 1310 (step 4237). The data exchange method 141 then creates a message object 110 containing the ad query (step 4238). The data exchange method 141 transmits

this message object 110 to the classified ad partner server 1302 (step 4239). When received by the classified ad partner server 1302, the message object 110 triggers a corresponding data exchange method 141 (step 4240). This data exchange method 141 uses the ad query to to query the classified ad partner server database 1301 for any ad listings satisfying the query (step 4241). The data exchange method 141 then returns the result set to the consumer program 12 (step 4242). If there are no ad listings that satisfy the query, the result set is a message object 110 reporting this. If there are ad listings that satify the query, the result set are the communications objects 110 representing the seller together with the component object 110 representing the seller's ad listing. This is advantageous because these communications objects 110 enable the buyer and seller to immediately establish their own communications relationship to consummate the sale. After the consumer program 22 receives the result set, it executes any receipt methods pertaining to the result set objects (step 4243). This includes the notification test (step 4244). If the there were no matching ads, the buyer may not desire notification. If there are matching ads, the buyer may desire different notification based upon the attributes values of the matching ads. For example, if a minivan meeting the buyer's query was listed at a price below a certain value, the buyer might desire to be paged immediately, whereas at a higher price the buyer may only wish to receive notification in the buyer's notification report (630, FIG. 13). The provider program 12 then executes any desired notification methods (step 4245). Again, this process could also incorporate authentication, payment, reporting, or other service object services.

Detailed Description Text (401):
The above classified ad monitoring process operates by storing and executing the classified ad query at the consumer program 22. In the same fashion that it can be more efficient to monitor a high-volume directory partner server 1302 using a user object index, it can be more efficient to monitor a high-volume classified ad partner server 1302 using a user object index that includes the buyer's stored queries. This also allows query result sets to be transmitted to to buyers via the push technique, as opposed to the pull technique illustrated above. The steps for implementing monitoring with a user object index with stored queries at a partner server are shown in FIG. 35. The first step is identical to step 4235 of FIG. 34B, where the data exchange method 141 of the classified ad service object 1310 saves the input form data as a query element or elements 143 (step 4261). The data exchange method 141 then creates a message object 110 containing the query element or elements 143 (step 4262). This message object 110 also contains such other communications object components as are necessary to create a user object 110 representing the buyer. This includes a data exchange method 141 for processing result sets produced at the classified ad partner server 1302. The data exchange method 141 of the classified ad service object 1310 next transmits this message object 110 to the classified ad partner server 1302 (step 4263). When received by the classified ad partner server 1302, the message object 110 triggers a corresponding data exchange method 141 (step 4264). This data exchange method 141 first creates a user object 110 representing the buyer in the classified ad partner server database 1301 (step 4265). Next the data exchange method 141 creates a communications object 110 of the component object type (812, FIG. 17) containing the ad query element or elements 143 (step 4266). This component object 110 is given a member association 110A with the buyer's composite communications object 110 (step 4267). In this manner the buyer can submit additional ad queries without needing to duplicate the user object 110 representing the buyer. User objects 110 and ad query component objects 110 can also be indexed for performance optimization. Next the data exchange method 141 uses the query elements 143 of the ad query component object 110 to create one or more scheduled event instances 117 in the classified ad partner server database 1301 (step 4268). As with stored queries in the consumer program 22, these scheduled event instances 117 can begin immediately and repeat at intervals or according to rules 140 specified by the consumer or the classified ad service provider. When activated, these scheduled event instances execute a data exchange method 141 on the classified ad partner

server 1302 (step 4269). This data exchange method 141 then executes the ad query (step 4270). When the query result set is returned, the data exchange method 141 calls another data exchange method 141 in the buyer user object 110 (step 4271). The buyer's data exchange method 141 processes the result set to determine if notification is desired by the buyer (step 4272). If so, the buyer's data exchange method 141 calls a data exchange method 141 in the classified ad partner server 1302 (step 4273). This data exchange method 141 creates a message object 110 containing the result set (step 4274) and transmits it to the consumer program 22 (step 4275). There the consumer program 22 receives the message object 110 and executes its receipt method 141 (step 4276). The consumer program 22 then executes any notification methods 141 specified by the buyer to control notification about classifed ad queries (step 4277).

Detailed Description Text (402):
The data exchange procedures illustrated here for sellers and buyers using a classified service object 1310 to automate interchange with a classified advertising database 1301 stored on a classified ad partner server 1302 can be generalized to any type of data that can be stored in a server database mutually accessible to providers and consumers on a communications network 3. This includes search services, news services, document retreival services, knowledgebases, discussion databases, and so on. The specific type of database, database server, or data exchange service is not a limiting feature of the invention. The only differences are the organization and format of the data stored on the server database and the queries and rules used to automate information interchange. These generalized steps are summarized in FIG. 36. The first step is to use composite and component communications object types (811, 812, FIG. 17) to represent organizational or topic structure in the data exchange partner server database 1301 (step 4291). This creates a metadata structure for the data stored in the database. Category objects discussed above and shown in FIG. 29A are one example of such a metadata structure. A second example is shown in FIG. 29B. This illustrates how composite and component communications objects can be organized as the main topic and response "threads" in a message database, discussion database, or knowledgebase. The main topic is a composite communications object type 1451. Each response to this main topic is shown as a first-level response thread object 1461, 1462. Each response to a first-level response is shown as a second-level response thread object 1471, 1472, 1473, 1474. Just as software object relationships can be used generally to model many real-world relationships, the association and aggregation relationships (shown in FIGS. 3 and 4) between composite and component communications objects can be used generally to model many metadata relationships. The first advantage to using communications objects 110 to represent the metadata structure of a partner server 1302 is that the structure is dynamic. Communications objects 110 can be added, deleted, and edited easily. Secondly, when these changes take place to the metadata structure, every provider and consumer affected by the change is updated and notified automatically.

Detailed Description Text (403):
Referring back to FIG. 36, the second step is to use communications object type definitions 144 and elements 143 to model the data and metadata stored within the larger metadata structures on the partner server (step 4292). The use of type definitions 144 and elements 143 to model data and metadata is explained in multiple sections above. A specific example is the use of notification elements and message elements (201, 211, FIG. 4) as discussed in the notification control section above. Again, the use of software objects allows such modelling to be applied broadly to many real-world database needs. The third step is to use data exchange service objects 1310 and message objects 110 to automate data exchange between the programs 12, 22 and the data exchange partner server 1302 (step 4293). As explained throughout this section and the data exchange control section, a combination of data exchange methods 141, data exchange rules 140, and data exchange elements 143 can be used to automate many kinds of data uploading and updating for providers and data monitoring, querying, and downloading for

consumers. The fourth step is using user object indexes representing the providers and consumers interacting with the data exchange partner server 1302 whenever transferring processing tasks from the programs 12, 22 to the partner server will increase data exchange or network efficiency (step 4294). This procedure is explained in the service object introduction section, the directory service object section, the distribution service object section, and illustrated in detail in this section and FIG. 34B. In particular, it allows data updates or query result sets to be transmitted via the push technique when it is more efficient than the pull technique. The fifth step is to use data exchange methods 141 and notification control at the programs 12, 22 to process message objects and query result sets (step 4295). This is shown in the final steps of FIGS. 34A, 34B, and 35. This allows providers and consumers to realize the maximize benefit of data exchange automation by not being notified of anything but the most highly relevant information, and then having complete control over the notification method. The sixth step is returning communications objects 110 in query result sets whenever it would increase the efficiency of data exchange or communications for the user (step 4296). This is illustrated in step 4242 of FIG. 34B, where the seller's communications object 110 is returned together with the seller's ad listing. This technique is particularly powerful when the communications object 110 returned is a category object 110 or a service object 1310 or granting the user access to a new set of communications objects 110 or partner servers 1302. The final step is using link control in communications objects 110 and category objects 110 to simplify and automate access to service objects 1302 and other category objects 110. This is discussed in the service object introduction section and the directory service object section. This technique spreads linking intelligence with every communications object 110 in a communications object system, making it as easy as possible for the users to gain access to the services of any type of partner server 1302.

Detailed Description Text (406):
The following explains the basic processes involved with the use of payment service objects 1310 and payment partner servers 1302. These are broken into several sets as shown in FIGS. 37, 38, and 39. The steps in the process of a merchant creating a payment account are illustrated in FIG. 37. The process begins with the merchant obtaining a copy of the payment service object 1310 if one is not already present in the provider database 11 (step 4401). When the merchant is ready to use the payment service object 1310 to set up a payment account, the merchant activates a data exchange method 141 in the payment service object 1310 (step 4402). This data exchange method 141 first generates a public/private key pair, either itself or by calling the services of an authentication service object 1310 (step 4403). Alternatively the data exchange method 141 can use an existing public/private key pair available from the authentication service object 1310. The private key is stored as an element 143 of the payment service object 1310 in the provider database 11 (step 4404). As with an authentication private key, this key may also be encrypted with a password known only to the user and not stored locally. The data exchange method 141 then queries the provider database 11 for the elements 143 necessary to create a payment account (step 4405). This process is explained in the data exchange control section. Because many of these elements 143 are commonly required items of data, such as the provider's name, contact data, financial account data, credit references, and so on, they will already be present in the provider database 11 and can be automatically accessed by the payment service object 1310. The data exchange method 141 then generates an account data input form (step 4406). The purpose of this form, as with most data exchange input forms, is threefold. First, it allows the merchant to confirm any data exchange rules 140 the merchant may have applied to the transfer of the merchant's sensitive financial data, such as bank account numbers or credit references. Second, it allows the merchant to confirm the accuracy of any other data to be transferred, such as contact data. Third, it allows the merchant to enter any specific new data required by the payment service provider. As explained in the data exchange control section, such new data can also be saved as elements in the provider database 11 for future

use. When the merchant submits the completed input form, the data exchange method 141 creates a account order (step 4407). The payment service object 1310 then calls an authentication method 141 in an authentication service object 1310 to encrypt the account order using the payment partner server's public key, stored in the payment service object 1310 as an element 143 (step 4408). The authentication method 141 then digitally signs the account order using the merchant's private key (step 4409). The merchant's private key may be stored in the authentication service object 1310 as an encrypted element 143, in which case the authentication method 141 may first require a password from the merchant for decryption. Alternatively the merchant's private key may be entered manually in some other way. The data exchange method 141 now creates a message object 110 containing the secure account order and the merchant's public key certificate, stored as an element 143 in the authentication service object 1310 (step 4410). Optionally this message object 110 may also contain such data as is necessary to create a user object 110 representing the merchant at the payment partner server 1302. The data exchange method 141 transmits this message object 110 to the payment partner server 1302 (step 4411). The payment partner server 1302 receives the message object 110 and executes a data exchange receipt method 141 (step 4412). This data exchange method 141 calls the same authentication service object 1310 to decrypt the secure account order using the payment partner server's private key, stored as an element 143 in the partner server database 1301 (step 4413). Next the authentication service object 1310 verifies the merchant's public key certificate signature using the authentication partner server's public key, stored in the authentication service object 1310 as an element 143 (step 4414). Finally the authentication service object 1310 verifies the merchant's signature on the account order using the merchant's public key (step 4415). Now the data exchange method 141 on the payment partner server 1302 can execute whatever steps are necessary to use the account order to create a merchant account (step 4416). In a preferred embodiment, the merchant account would be represented by a user object 110 in the payment partner server database 1301. When finished, the data exchange method 141 creates a merchant account certificate consisting of the merchant's account number, the merchant's provider UID, and whatever other data the payment service provider wishes to include in the certificate, such as a timestamp, account type identifiers, payment partner server identifiers, and so on (step 4417). (The merchant account certificate may also be encrypted if desired using a single key; decryption will only be necessary at the payment partner server.) The data exchange method 141 then calls an authentication method 141 in the authentication service object 1310 to digitally sign the merchant account certificate using the payment partner server's private key (step 4418). Next the data exchange method 141 creates a message object 110 containing the signed merchant account certificate (step 4419). The data exchange method 141 transmits this message object 110 back to the payment service object 1310 in the merchant's provider program 12 (step 4420). There the provider program 12 receives the message object 110 and executes the original data exchange method 141 of the payment service object 1310 (step 4421). This data exchange method 141 first calls an authentication method 141 in the authentication service object 1310 to verify the signature of the merchant account certificate using the payment partner server's public key (step 4422). Then the data exchange method 141 stores the merchant account certificate in the provider database 11 as an element 143 of the payment service object 1310 (step 4423). Lastly the data exchange method 141 calls any notification methods desired by the merchant for notification of the merchant account certificate receipt (step 4424). This completes the process of setting up a secure payment account for the merchant.

Detailed Description Text (407):
To begin using this account with customers, the merchant includes the merchant account certificate and a link component object 110 from the payment service object 1310 in any communications object 110 where the merchant wishes to use payment services. The payment service object 1310 can then be called by any data exchange method 141 in the merchant's communications object 110. The merchant can indicate the services of such payment service objects 1310 by using the names or logos of

the appropriate credit cards, debit cards, and so on in a product ordering input form, for example. When a customer chooses one of these options and submits a data exchange input form, the payment service object 1310 is used automatically. The steps in this process are shown in FIG. 38. First the data exchange method 141 creates a purchase order consisting of the data from the input form together with the merchant account certificate (step 4441). Next the data exchange method 141 queries to see if the payment service object 1310 is present in the customer's consumer database 21 (step 4442). If not, the data exchange method 141 uses the payment service object's link component object 110 to download the payment service object 1310 (step 4443). The payment service object's receipt method 141 will then initiate the process to create a customer account (step 4444). This process is identical to the merchant payment account creation process shown in FIG. 37, except the final result is that the customer is issued a customer account certificate stored in the consumer database 21 as an element 143 of the payment service object 1310. If the payment service object 1310 was present in the consumer database 21 in step 4441, the data exchange method 141 calls a version monitoring method 141 to see if the version is current (step 4445). This version monitoring method 141 compares the version value of the payment service object 1310 with the version value stored in the link component object 110 of the merchant's communications object 110. Version monitoring is explained in the data exchange control section above. If the version is not current, the data exchange method 141 executes the update method 141 of the payment service object 1310 to download the current version (step 4456). Once the current version of the payment service object 1310 is present in the consumer database 21, the data exchange method 141 in the merchant's communications object 110 calls a data exchange method 141 in the payment service object 1310 to continue the transaction (step 4457). This data exchange method 141 calls an authentication method 141 in an authentication service object 1310 to encrypt the purchase order using the payment partner server's public key, stored in the payment service object as an element 143 (step 4458). The authentication method 141 also digitally signs the purchase order using the customer account certificate private key (step 4459). As described above, this key may be stored as an encrypted element 141 in the payment service object 1310 and require a password from the customer to decrypt. Alternatively the customer may supply the key manually in some other way. Next the data exchange method 141 creates a message object 110 containing the secure purchase order and the customer account certificate (step 4460). The data exchange method 141 transmits this message object 110 to the payment partner server 1302 (step 4461). The payment partner server 1302 receives the message object 110 and executes its receipt method 141, which is either the same data exchange method 141 or another data exchange method 141 residing on the payment partner server 1302 (step 4462). This data exchange method 141 calls an authentication method 141 in the authentication service object 1310 to verify the customer's signature on the secure purchase order using the customer account certificate public key (step 4463). Next the authentication method 141 decrypts the purchase order using the payment partner server's private key (step 4464). Finally the authentication method 141 verifies the merchant's signature on the merchant account certificate using the payment partner server's private key (step 4465). Now a data exchange method 141 on the payment partner server 1302 can carry out the purchase order transaction using the verified purchase order data, the verfied customer account certificate, and the verified merchant account certificate (step 4466). This may involve any sequence of steps between the payment partner server 1302 and other payment servers or data processing systems, such as the consumer's bank or credit clearinghouse, a credit card processor, a cybercash server, and so on. When the transaction has been completed, the data exchange method 141 creates a unique receipt number stored as an element 143 in the payment partner server database 1301 (step 4467). This receipt number can now be used to verify the transaction with both the customer and the merchant.

Detailed Description Text (408):
From this point the receipt acknowledgment process can take several paths. The payment partner server 1302 can return receipt acknowledgments to both the consumer

program 22 and the provider program 12. Each of these programs can in turn send receipt acknowledgments to the other to complete full three-way acknowledgment. Alternatively the payment partner server 1302 can send a receipt acknowledgment to the customer's consumer program 22, which can in turn send a receipt acknowledgment to the merchant's provider program 12, or vice versa. In all cases the steps in sending secure receipt acknowledgment messages are similar. The steps in the process of the payment partner server 1302 sending a receipt acknowledgment message to the customer's consumer program 22 are shown in FIG. 39. First a data exchange method 141 on the payment partner server 1302 creates a purchase receipt (step 4471). The purchase receipt includes the unique receipt number plus any other relevant data, such as timestamp, the payment partner server UID, bank certification numbers, and so on. Next the data exchange method 141 calls an authentication method 141 in an authentication service object 1310 to encrypt the purchase receipt using the customer account certificate public key (step 4472). This step is optional if the purchase receipt does not contain any sensitive information. The authentication method 141 then signs the purchase receipt with the payment partner server's private key (step 4473). The data exchange method 141 creates a message object 110 containing the purchase receipt (step 4474). The data exchange method 141 transmits this message object 110 to the payment service object 1310 at the consumer program 22 (step 4475). There the consumer program 12 receives the message object 110 and executes the original data exchange method 141 of the payment service object 1310 (step 4476). This data exchange method 141 first calls an authentication method 141 in the authentication service object 1310 to verify the signature on the purchase receipt using the payment partner server's public key (step 4477). If the purchase receipt has been encrypted, the authentication method 141 decrypts it using the customer account certificate private key (step 4478). Then the data exchange method 141 stores the purchase receipt in the consumer database 21 as an element 143 of the payment service object 1310 (step 4479). This makes the purchase receipt available to the payment service object 1310 and the merchant communications object 110 for use in any further transactions or correspondence involving this transaction, such as a return or exchange. Finally the data exchange method 141 executes any notification methods 141 desired by the customer for notification about the receipt acknowledgment (step 4480).

Detailed Description Text (412):
Shared access to the methods 141 of a reporting service object 1310 is particularly efficient for gathering statistics and metadata for a large population of communications objects 110. This is because statistics and metadata for a large number communications objects 110 from a large number of providers can accumulated in the databases 11, 21 and then transmitted using a small number of message objects 110 to one or more reporting partner servers 1302. The same reporting service object 1310 or a linked reporting service object 1310 can then be used by the providers to monitor the aggregated reports at the reporting partner server 1302. This can be done using data exchange methods 141 running queries against a reporting partner server 1302 in the same fashion as explained in FIGS. 34B and 35 for classified ad buyers.

Detailed Description Text (418):
A feedback service object type (844, FIG. 17) is another specialized data exchange service object that works in conjunction with a feedback partner server 1302 to aggregate and report feedback from users of any communications object 110 across a communications object system. Feedback service objects 1310 combine the functions of directory service objects, data exchange service objects, and reporting service objects to aggregate feedback from multiple communications object system users across different categories of communications objects 110. Feedback service objects 1310 are another excellent example of a polymorphic service object because the same feedback service object 1310 that is used to provide feedback from one communications object system user (called the "feedback provider") can be used to access or monitor that feedback by another communications object system user (called the "feedback consumer").

Detailed Description Text (419):
Feedback service objects 1310 can most easily be understood as an extension to the functionality of category objects 110. Category objects 110 are explained in the directory service object section above and shown in FIG. 29A. In particular, communications objects 110 listed in a directory partner server 1302 can include a link component object 110 to each category object 110 with which the communications object 110 is associated. This allows users of a communications object system to quickly and easily obtain category objects 110 to check directory partner servers 1302 for other communications objects 110 associated with the category object 110. This process extended into a powerful feedback system with three enhancements. First, feedback attributes and value choices can be added to category objects 110. This permits users of a communications object 110 can be solicited for feedback specific to that particular category of communications objects 110. Second, report processing capabilites can be added to directory partner servers 1302. This permits feedback data to be aggregated into reports of high value to communications object system users. Third, the feedback attributes of the category objects 110 can be used by feedback consumers to create queries to a feedback partner server 1302. This allows feedback consumers to be automatically notified of new or changed communications objects 110 that meet the user's specific interest criteria.

Detailed Description Text (420):
The steps in these processes are very similar to those described for classified ad category objects in the data exchange service object section above and in FIGS. 34A, 34B, and 35. The following explains the basic processes involved with the use of feedback service objects 1310 and feedback partner servers 1302. Both the feedback provider and feedback consumer will be illustrated as users of the consumer program 22. The steps in the process of a feedback provider submitting feedback input are illustrated in FIG. 41A. We will assume the feedback provider already has obtained the communications object 110 upon which the feedback will be given. The process begins with the feedback provider executing a feedback link method 141 in the communications object 110 (step 4601). The presence of a feedback link method 141 for feedback can be shown by the use of a feedback hyperlink or hypergraphic on a page 142 of the communications object 110. Alternatively, if feedback services are implemented as a global function of a communications object system, this feedback link method 141 can be available as a system method 141 to all communications objects 110. In this case feedback services could be initiated through a feedback hyperlink or hypergraphic on the selected object form (611, FIG. 13), on any selected page form (612, FIG. 13), or on a global toolbar in the consumer program 22. Once executed, the feedback link method 141 queries the consumer database 21 for the linked feedback category object 110 (step 4602). If the link method 141 is linked to more than one feedback category object 110, the link method 141 can present the feedback provider with an input form to select the desired feedback category object 110. If the feedback category object 110 is not present, the link method 141 uses link control to download the feedback category object 110 (step 4603). Next a link method 141 in the feedback category object 110 queries the consumer database 21 for the linked feedback service object 1310 (step 4604). If the feedback service object 1310 is not present, the link method 141 uses link control to download the feedback service object 1310 (step 4605). Now a data exchange method 141 in the feedback service object 1310 generates a feedback input form (step 4606). This input form consists of the category attribute and value choices obtained from the feedback category object 110. This is identical to the input form process described in step of 1704 of FIG. 34A for creating a classified ad listing. The feedback attributes and value choices for any feedback category object 110 are determined by the feedback service provider. To continue the example used in the classified ad service description, the feedback input form for a feedback category object 110 representing minivans might include attributes for dealer satisfaction, fit and finish, gas mileage, maintenance costs, repurchase plans, and so on. The appropriate value choices for each of these attributes would be displayed as drop-down lists, radio buttons, and so on. When the feedback

provider submits the completed input form, the data exchange method 141 first saves the feedback data as a feedback element 143 of the feedback service object 1310 (step 4607). This permits the feedback provider to easily recall and modify the feedback data as his/her feedback changes. Next the data exchange method 141 creates a message object 110 containing the feedback data from the input form and the UID of the target communications object 110 (step 4608). The data exchange method 141 then transmits this message object 110 to the feedback partner server 1302 (step 4609). When received by the feedback partner server 1302, the message object 110 triggers a corresponding data exchange method 141 (step 4610). This data exchange method 141 can aggregate and process the feedback data in the feedback partner server database 1301 as proscribed by the feedback serve provider (step 4611). Feedback data aggregation and processing is described below. The data exchange method 141 can also perform any kind of aggregation or statistical analysis required to produce the reports the feedback service provider wishes to offer feedback consumers. The attributes of these reports must match the feedback report query attributes of a feedback category object 110 as described below.

Detailed Description Text (421):
Feedback data can be aggregated by a feedback partner server 1302 in several ways. One approach is to save and index feedback data by the UID of the feedback provider. In this approach the feedback partner server 1302 maintains records of the feedback data from each feedback provider. This allows the feedback partner server 1302 to produce accurate feedback statistics reports over time. Another approach is to aggregate feedback using counters. In this approach the feedback partner server 1302 does not need to maintain a record from each feedback provider. Instead the feedback partner server 1302 increments a counter for each feedback message object 110 received from a feedback provider. The accuracy of this counter is maintained in the following manner. The first time a feedback provider uses a feedback category object 110 to send feedback data, the full set of feedback data is transmitted in the feedback message object 110 as described in step 4608 of FIG. 41A. This feedback data is saved as an element 143 of the feedback category object 110 the consumer database 21 as described in step 4607 of FIG. 41A. The steps required for subsequent changes to the feedback are shown in FIG. 41B. First the feedback provider executes the feedback link method 141 as in step 4601 of FIG. 41A (step 4631). Because the feedback provider has already submitted feedback, the feedback category object 110 and feedback service object 1310 are present in the consumer database 21. Thus the feedback link method 141 can directly execute a data exchange method 141 of the feedback service object 1310 (step 4632). This data exchange method 141 reads the saved feedback element 143 of the feedback category object 110 (step 4633). The data exchange method 141 uses this feedback data to generate an input form for the feedback provider to edit the feedback data (step 4634). When this input form is submitted, the data exchange method 141 saves the new feedback data as a new version of the feedback element 143 (step 4635). Versions of the feedback data element can be controlled using data archive control as explained above. The data exchange method 141 next calculates the differentials between the new feedback data and the old feedback data (step 4636). For example, a minivan owner originally rated dealer service at an 8 on a scale of 1 to 10. The minivan owner subsequently had a poor dealer service experience. The minivan owner then edits the feedback data to rate the dealer service at a 2. The data exchange method 141 would calculate the differential as a minus 6. Now the data exchange method 141 creates a message object 110 containing the feedback differential data, the UID of the target communications object 110, and a "RevisedFlag" element 143 set to TRUE (step 4637). The data exchange method 141 transmits this message object 110 to the feedback partner server 1302 (step 4638). When received by the feedback partner server 1302, the message object 110 triggers a corresponding data exchange method 141 (step 4639). This data exchange method 141 uses the value of the RevisedFlag element 143 to process the feedback data as incremental data and not new data. The data exchange method 141 then uses the differentials in the feedback data to adjust the feedback counters (step 4640). Revised feedback data will not increment a "Total Feedback Providers" counter, so feedback consumers can see an

accurate report at of the total aggregated feedback at any point in time and the total number of feedback providers who have contributed to this feedback data. Feedback data counters allow the maintenance of feedback data to be distributed throughout a communications object system, making it feasible to centrally aggregate feedback for a large number of communications objects 110 from a large population of feedback providers. Feedback data counters also make it easy to do anonymous reporting, as no provider UIDs must be tracked at the reporting partner server 1302. Alternatively, if feedback data is saved and indexed at the reporting partner server 1302, anonymous reporting can be accomplished using the anonymous reporting key technique as described in the reporting service object section above and illustrated in FIG. 40.

Detailed Description Text (422):
The steps in the process of a feedback consumer accessing and monitoring feedback are identical to those of a classified ad buyer accessing and monitoring classified ad listings as explained in the data exchange service object above and illustrated in FIGS. 34B and 35. As with directory category objects 110, a feedback consumer can obtain a feedback category object 110 either directly from a feedback partner server 1302 or by using a link component object 110 in any communications object 110 associated with that feedback category object 110.

Detailed Description Text (423):
The value of feedback data can vary enormously with the experience and expertise of the feedback provider. This is particularly true for feedback on topics requiring specialized knowledge or expertise, such as academics, law, medicine, technology, and so on. For this reason feedback services can also be applied to feedback providers. This can be accomplished using a feedback partner server 1302 by linking feedback category objects 110 to user objects 110 representing each of the feedback providers. The attributes of a feedback category object 110 representing a feedback provider might include level of expertise, level of credibility, level of decision-making ability, and so on. By aggregating feedback data on feedback providers, a feedback partner server 1302 is able to offer even more useful feedback reports to feedback consumers. This is because feedback queries can select feedback data using on the attributes or "ratings" of the feedback providers. An example is a feedback partner server 1302 which collects feedback data on communications objects 110 representing automobiles. A feedback consumer can create a query for only those communications objects 110 representing minivans with a sticker price of less than $20,000 which also had overall quality rating of 7 or higher on a scale of 1 to 10 from feedback providers whose expertise level was rated by other feedback providers to also 7 or higher on a scale of 1 to 10. Another example applies to response thread objects (FIG. 29B) in a topic discussion database. Here a feedback consumer can use a topic feedback category object 110 to monitor the response thread objects 110 contained by a discussion topic 110. A query can notify the feedback consumer only of new response thread objects posted by providers with an expertise rating of 7 or higher on a scale of 1 to 10. A feedback consumer can also ask for feedback provider ratings to be factored into feedback data reports. An example would be a report on recommended minivans where the feedback data from feedback providers with an expertise rating of 8 or higher on a scale of 1 to 10 was weighted twice as heavily as feedback data from feedback providers with a rating lower than 8.

Detailed Description Text (426):
As with directory services, feedback services can be employed for a wide variety of purposes on a communications object system. This includes product and service rating services, political office ratings, employee performance feedback, discussion group participation, personal references, and so on. The particular feedback service is not a limiting feature of the invention.

Detailed Description Text (429):
It has been explained how in an embodiment of the present invention the functions of the provider and consumer programs 12, 22 and databases 11, 21 can be combined

because they use identical database structures and similar operations. In another embodiment of the present invention, the functions of either or both the programs 12, 22 and databases 11, 21 can be combined with a partner server 1302 and a partner server database 1301. This is again because identical database structures and similar operations are used. All programs can also employ the same HTML and HTTP interface operations as described above. This means that a communications object system user may fully access the capabilities of a provider program 12, a consumer program 22, and a partner server 1302 all from a single web server 32 using a single web browser 50.

Detailed Description Text (430):
One of the additional benefits of combining the provider program 12 with a distribution server 32 is that providers do not have to transmit new and updated communications objects 110 to a separate distribution server 32 for distribution via the pull technique. Nor do they require the services of a distribution service object 1310. Rather pull updating from a consumer program 22 can take place directly from the combined provider program 12 and partner server 1302. This saves time and reduces the potential for transmission errors. A provider is also able to more easily apply distribution control by specifying distribution control methods 141 directly in the combined database 100.

Detailed Description Text (434):
The programs 12, 22, and 1302 or any combination thereof can accommodate multiuser operation. In all cases this can be accomplished by employing the user object type (816, FIG. 17). The data structures for user objects 110 are shown in FIG. 6B. Like all other communications objects 110, user objects 110 have a system ID attribute that uniquely identifies them within the database 100. The provider and consumer relationship between user objects 110 and other communications objects 110 uses a relationship association class 111 of the association 110A. One attribute of the relationship class 111 is a logical value "ProviderFlag". If a user is a provider of a communications object 110, the ProviderFlag value is TRUE. If the ProviderFlag value is FALSE, the user is a consumer of the communications object 110. The relationship class 111 also has an attribute NewFlag which is employed in user object indexes as described above. The relationship class 111 may have other attributes such as "PrivilegeLevel" that govern access control to operations such as editing the communications object, forwarding the communications object, and so on. Access control will be discussed below.

Detailed Description Text (435):
User objects 110 can represent individual communications object system users or groups of users. User group objects 110 function similarly to e-mail aliases in an e-mail system. Groups can be nested by creating composite user objects 110 and components user objects 110. User group objects 110 can also have their own distinct attributes used to control the communications functions and privileges of the group.

Detailed Description Text (436):
Multiuser operation is beneficial in the programs 12, 22 or a program combining their operation because it allows a single database 100 to be shared by multiple users. A separate instance of the global preferences class 103 can be associated with each user object 110. In a multiuser database 100, multiple user objects 110 can have a consumer relationship association 111 with single instance of communications object 110. This saves disk space and increases overall system efficiency. In this case each consumer can maintain separate preferences using separate element preferences instances 147 associated with the consumer's user object 110. Users can also share preferences by having an association to the same element preference instance 147. Access control rules 140 can be used to govern editing rights to shared element preference instances 147. Access control rules will be discussed below.

Detailed Description Text (437):
In a multiuser database 100, a user object 110 can have a consumer relationship with a communications object 110 to which another user object 110 has a provider relationship. This has several very important benefits. To begin with, no instance of the recipient class 120 nor the acknowledgement association 121 is needed. Both can be replaced entirely by the relationship associations 111. Secondly, no communications object distribution routine is necessary. When the user object 110 representing a provider (called the "provider user object") and the user object 110 representing a consumer (called the "consumer user object") are both present in a multiuser database 100, a communications object 110 can be "pushed" to a consumer simply by the provider creating a new association between the communications object 110 and the consumer user object 110. A communications object 110 can be "pulled" by a consumer just by the consumer creating a new association between the communications object 110 and the consumer user object 110. In both of these cases, creation of the new association triggers a "new object reception rule" 140 in the database 100. This rule takes the place of the new object reception routine in a separate consumer program 22 and executes steps 703-708 of FIG. 15. Updates to a communications object 110 by the provider can also be "transmitted" to all associated consumers via the operation of the standard update association rule 140 operating throughout the database 100. This operation of this rule takes the place of the update object reception routine (FIG. 10B) by executing steps 721-731 of FIG. 15. This rule 140 only applies to consumer relationship associations, i.e. where the ProviderFlag value is FALSE.

Detailed Description Text (438):
Finally, in a multiuser database 100, multiple user objects 110 can also have a provider relationship with a single communications object 110. This is referred to as multiuser editing. Multiuser editing of communications objects 110 is advantageous in a communications object system for the same reasons multiuser database sharing is advantageous in many business applications. Just as two or more individuals can need the ability to read or edit same data, two or more individuals can need to communicate about the same subject or topic through the same "channel". In many multiuser database environments, including network file systems, database access and editing rights are controlled using access control lists. This same principle can be applied to a communications object system through the use of access control elements 143, access control methods 141, and access control rules 143. Collectively these are referred to as access control components. Access control elements 143 are special elements 143 included in a communications object 110 in order to define the editing rights which the original communications object provider wishes to grant to other providers. Access control methods 141 and access control rules 140 act in conjunction with access control elements 143 to enforce these rights. Access control is an extension of data exchange control, discussed in the data exchange control section above. Access control components are a unique advantage of a communications object system because they can be contained within the communications object 110 which they govern. Thus they can be distributed and enforced throughout a communications object system. Access control rights can also be governed using user group objects 110. In this capacity user group objects 110 function similarly to access control groups used in many computer network environments to govern file and resource access.

Detailed Description Text (439):
As in any multiuser database, simultaneous editing of the same data field or record by different users can result in conflicts. Many multiuser database record locking or data conflict resolution techniques have been developed to solve this problem, including rules based on time precedence, user priority, location priority, data types, and so on. This is referred to as concurrency control. In a multiuser communications object system database 100, concurrency control can be applied using concurrency elements 143, concurrency methods 141, and concurrency rules 140. Collectively these are referred to as concurrency control components. Concurrency control can be applied in one or more communications object system databases 100 in

the same manner as access control. The specific concurrency control rules or techniques employed are not a limiting feature of the invention.

Detailed Description Text (440):
In a communications object system, multiuser editing applies to three situations. The first is single users operating different single-user installations of the combined programs 12, 22 on different computers 1, 2. The second is different users operating the same multiuser installation of the combined program 12, 22. This could be on a single central computer accessible over a local area network, or on a distributed database available over a wide area network. The third is a combination of the first two. In the first situation, multiple users can edit the same communications object 110 through the use of message objects 110. The basic steps involved with this form of multiuser editing are illustrated in FIG. 42A. The process begins with the first provider of a communications object 110 specifying the access control the provider wishes to apply to a communications object 110 (step 4701). This is done by specifying access control components and choosing appropriate access control values. Next the first provider transmits the communications object 110 to a recipient (step 4702). The consumer program 22 of the recipient receives the communications object 110 and stores it in his/her consumer database 21 (step 4703). The recipient next performs an editing operation on the communications object 110 (step 4704). These editing operations will be constrained by the access control components of the communications object 110. For example, the provider of a communications object 110 representing a discussion topic may grant recipients the right to add response thread objects 110 representing responses, such as is shown in FIG. 29B. However this provider may not grant recipients the right to edit the name, description, or message in the original topic object 110. The completion of an editing operation by the recipient triggers a data exchange control rule 140 which executes a data exchange method 141 of the communications object 110 (step 4705). In this way the data exchange control rule 140 serves the same purpose as the update association rule 140 in a database 100. The data exchange method 141 creates a message object 110 containing the changes to the communications object 110 (step 4706). The data exchange method 141 then transmits the message object 110 to the combined program 12, 22 of the first provider (step 4707). When the message object 110 is received, its receipt method 141 is executed (step 4708). The receipt method 141 saves the edits to the communications object 110 (step 4709). Conflicts in edits by two or more users are handled by concurrency control as discussed above. This save operation results in the same set of operations as an edit operation in the provider program 12 as shown in FIGS. 10A and 10B. These changes will now be distributed to all recipients of the communications object 110 using distribution control as described in the distribution control section above. Lastly, the receipt method 141 executes any notification methods 141 assigned by the first provider (step 4710).

Detailed Description Text (442):
The steps involved with multiuser editing using distribution control components are shown in FIG. 42B. The process begins with the first provider of a communications object 110 specifying both the access control and the distribution control the provider wishes to apply to a communications object 110 (step 4731). This is done by specifying access control components and values and distribution control components and values. Next the first provider transmits the communications object 110 to the recipients on the distribution control list (step 4732). The consumer program 22 of the recipient receives the communications object 110 and stores it in his/her consumer database 21 (step 4733). The recipients next performs an editing operation on the communications object 110 (step 4734). The completion of an editing operation by a recipient triggers a data exchange control rule 140 which executes a data exchange method 141 of the communications object 110 (step 4735). The data exchange method 141 creates a message object 110 containing the changes to the communications object 110 (step 4736). The data exchange method 141 then transmits the message object 110 to each of the recipients on the distribution control list (step 4737). When the message object 810 is received by each

recipient, its receipt method 141 is executed (step 4738). The receipt method 141 saves the edits to the communications object 110 (step 4739). Conflicts in edits by two or more users are handled by concurrency control as discussed above. This save operation results in the same set of operations as an edit operation in the provider program 12 as shown in FIG. 10A. However, because the changes have already been distributed to the distribution control list, this is an exception to the update association rule 140 and does not trigger the update association routine shown in FIG. 10B. Lastly, the receipt method 141 executes any notification methods 141 assigned by each recipient (step 4740).

Detailed Description Text (443):
The second multiuser editing situation applies when the combined programs 12, 22 are operated as a multiuser system and the providers and consumers involved are all users of this system. In this case multiuser editing operates in the same manner as record sharing in a typical multiuser database environment. Each communications object 110 is a record created by the first provider in the database 100, and the access control components of the communications object 110 act as the access control rights for other users of the database 100. In a multiuser database 100, access control rights can also be governed by the attributes of a relationship association (111, FIG. 6B). In this manner a provider can grant different access control rights to different user objects 110 or user group objects 110 representing other providers. Distribution control within a multiuser database 100 is not necessary because, as explained above, distribution in a shared database 100 is accomplished through the operation of the update association rule 140.

Detailed Description Text (444):
The third multiuser editing situation is a combination of the fist two, i.e. users spread across both single-user installations and multiuser installations of the programs 12, 22. This operates as a special case of the first situation. In this case, distribution control lists can contain special entries representing multiple users at a multiuser installation. These special entries can consist of nested elements 143 representing the UID of the multiuser database 100 and the UIDs of each individual user object 110. Alternatively they can contain nested composite and component objects 110 representing the multiuser program 12, 22 and the individual user objects 110. User group objects 110 can also be employed for this purpose. In this manner only one communications object or communications object update transmission needs to take place to each multiuser database 100. The receipt method 141 of the communications object 110 or message object 110 can then create or update the relationship associations 111 to each user object 110 in the multiuser database 100.

Detailed Description Text (448):
Because a communications object system permits providers and consumers to control any type of communications over any type of communications network 3, it can be particularly useful for coordinating communications relationships that take place over multiple communications networks. A simple example is a Internet-based fax request system. Such a system allows users to request fax documents using a web server 50 and have them transmitted to the user as fax documents via a telephone network. Such a system can easily be automated using communications object system. This can be accomplished using data exchange methods 141 included directly in a communications object 110, or it can be done using a data exchange service object 1310. The latter permits fax services to easily be shared among many communications objects 110. The steps for automating a fax request system using a fax service object 1310 are shown in FIG. 43. The process begins with the consumer obtaining the communications object 110 offering fax request services (step 4801). Next the consumer selects the hyperlink or hypergraphic representing a fax request page within the communications object 110 (step 4802). This executes a link method 141 which checks to see if the linked fax service object 1310 is present in the consumer database 21 (step 4803). If not, it uses link control to download the fax service object 1310 (step 4804). Next a data exchange method 141 in the fax service

object 1310 generates an input form (step 4805). This input form allows the consumer to select the fax documents the consumer would like to receive as well as the target fax number. Note that the fax service object 1310 can query the consumer database 21 for elements 143 with a type definition 144 of "FaxNumber" in order to automatically present such a list. The fax service object 1310 can also prompt the user to enter new fax numbers if none are present. When the consumer submits the input form (step 4806), the data exchange method 141 first saves any new fax number elements 143 as well as the consumer's preferences about the last-used fax number, the most-used fax number, and so on (step 4807). The data exchange method 141 then creates a message object 110 (step 4808). This message object 110 contains the indentification data for the requested fax documents (which could be the UIDs of the fax documents if they are stored as elements 143 on the fax partner server 1302). The message object 110 also contains the selected fax number. The data exchange method 141 transmits the message object 110 to the fax partner server 1302 (step 4809). When the message object 110 is received by the fax partner server 1302, it executes a corresponding data exchange method 141 (step 4810). This data exchange method 141 uses the fax request IDs and fax number to transmit the requested fax documents via the telephone network (step 4811). Note that if the fax partner server 1302 cannot successfully transmit the fax documents, the fax partner server 1302 can also send an acknowledgment message object 110 back to the consumer via the communications object system.

Detailed Description Text (449):
A more complex example is the coordination of package deliveries over a physical communications network such as a postal network. This process uses account certificates as described in the payment service object section. The steps in this process are shown in FIG. 44. The process begins with the consumer obtaining the communications object 110 of the package recipient (step 4901). Next the consumer selects a hyperlink or hypergraphic representing a physical package delivery option within the communications object 110 (step 4902). Such an option might be represented by hypergraphics of the logos of common delivery services. This executes a link method 141 which checks to see if the linked physical delivery service object 1310 is present in the consumer database 21 (step 4903). If not, it uses link control to download the physical delivery service object 1310 (step 4904). Next a data exchange method 141 in the physical delivery service object 1310 generates an input form (step 4905). This input form allows the consumer to select the desired delivery options. This includes the type of delivery required, whether delivery pickup is needed, payment options, and so on. Note that the consumer does not need to enter any information pertaining to the delivery attributes of the recipient. The data exchange method 141 is able to obtain all such attributes from the account certificate included in the recipient's communications object 110. If the consumer also has a account certificate, the data exchange method 141 can use this data automatically as well. If the consumer does not have an account certificate, the data exchange method 141 can use the input form to prompt the consumer for the necessary account information. When the consumer submits the input form (step 4906), the data exchange method 141 saves any new account data or preferences as elements 143 of the physical delivery service object 1310 (step 4907). The data exchange method 141 then creates a message object 110 containing the recipient account certificate, the sender account certificate, the delivery options selected, and any other pertinent data, such as a timestamp (step 4908). The data exchange method 141 transmits the message object 110 to the physical delivery partner server 1302 (step 4909). When received by the physical delivery partner server 1302, the message object 110 executes a corresponding data exchange method 141 (step 4910). This data exchange method 141 processes the delivery order (step 4911). This can include obtaining a delivery number, initiating the package delivery pickup by the physical delivery service provider, and any other steps necessary to initiate the delivery. When complete, the data exchange method 141 creates a acknowledgment message object 110 containing the delivery number together with any other acknowledgment data, such as the delivery pickup confirmation message (step 4912). The data exchange method 141 transmits this message object 110

to the physical delivery service object 1310 at the originating consumer program 22 (step 4913). When received at the consumer program 22, the message object 110 executes the originating data exchange method 141 of the physical delivery service object 1310 (step 4914). The data exchange method 141 first saves the delivery number and any other pertinent data as logged event 118 (step 4915). This allows it to be referenced for any future actions involving this delivery. Next the data exchange method 141 executes any notification methods 141 assigned by the consumer to delivery acknowledgment messages (step 4916). The data exchange method 141 then calls a print method 141 to print a delivery label (step 4917). The data exchange method 141 tests to see if delivery monitoring was requested in by the input form submitted in step 4906 (step 4918). If so, the data exchange method 141 carries out the monitoring process (step 4919). This monitoring process is identical to that performed by classified ad service objects 1310 as shown in FIG. 34B. Alternately, if the physical delivery partner server 1302 offers monitoring via a user object index and stored queries, the necessary user object data and query data can also be contained in the message object 110 created in step 4910. Monitoring using stored queries is shown in FIG. 35.

Detailed Description Text (452):
Schedule coordination among any group is a fundamental challenge in communications. Scheduled events and schedule changes must be communicated to everyone in the group or else the group will not function. A communications object system can solve many widespread scheduling problems using a special type of communications object called a schedule object. Schedule objects are shown as class 817 of FIG. 17. Schedule objects 110 represent real-world events associated with any other communications object 110. Like all other communications objects 110, schedule objects 110 can contain schedule elements 143, schedule methods 141, and schedule rules 140 used to control scheduling operations. Collectively these are referred to as schedule control components. These all function as special cases of data exchange control, discussed above. Schedule objects 110 can also be nested as composite and component objects (811, 182, FIG. 17). This permits a composite schedule object 110 to contain multiple component schedule objects 110. An example would be a composite schedule object representing a multi-day conference, with component schedule objects representing the individual seminars at the conference. Like any communications object 110, schedule objects 110 can be distributed via push and pull and contain their own update methods. This is particularly relevant to schedule coordination because any changes to a schedule object 110 can be transmitted to all consumers associated with that schedule object 110 object. Using notification control, each user can also control exactly how he/she is notified of these changes. Schedule objects 110 can be maintained in any database 100, whether single-user or multiuser. Schedule objects 110 are particularly useful for managing group schedules in a multiuser database 100. This is because schedule objects 110 can easily be associated with user objects 110 or user group objects 110 to create and maintain scheduling relationship associations 111. Alternatively, communications object system programs can handle scheduling requests through an API with an external scheduling program or database. A communications object system API is discussed further below.

Detailed Description Text (454):
Schedule objects 110 can be employed to solve a wide variety of common scheduling problems. One example is the universal problem of "telephone tag". In this example, schedule objects 110 are employed at a distribution server 32. Any type of distribution server 32 can be used, but in a preferred embodiment, a combination of the programs 12, 22 and a distribution partner server 1302 is used. This allows message objects 110 to be transmitted and received directly from the distribution partner server 1302 using a direct transmission protocol such as HTTP. This is faster than a store-and-forward protocol like Internet SMTP e-mail, however the latter can also be used if the scheduling process is not time-sensitive. To enable telephone call coordination, the provider first adds one or more scheduling elements 143, methods 141, and rules 140 to any communications object 110 in which

the provider wishes to offer scheduling control. The provider can also add schedule control using a scheduling component object 110. The provider also maintains his/her current schedule by adding and maintaining schedule objects 110 to the provider database 11. Thus the provider's current set of schedule objects 110 will indicate the present readiness of the provider to receive a phone call; the current phone number at which the provider should be reached; the options for notifying the provider about the call request; and the options for scheduling a phone call with the provider at some future time if the provider is not available immediately.

Detailed Description Text (456):
If the test in step 5006 determined the provider was available to take the call, the scheduling method 141 next checks the provider's scheduling preferences to see if the provider wishes to confirm phone call requests (step 5010). If so, the schedule object 110 calls the provider's notification method 141 for confirming phone call requests (step 5011). Notification control gives a provider rich control over such requests. Notification processing can take into account all of the data included in the call request message object 110 plus all of the data present in the combined programs 12, 22. This includes the consumer's identity, the presence or absence of the consumer UID in the provider database 21, the priority of a call, its expected length, and so on. By processing this data, a notification method 141 can produce any type of notification the provider desires. For purposes of this example, we will assume that this notification method generates an input form to obtain the provider's response. This form displays the data explained above. An option to initiate a phone call immediately from the provider back to the consumer could also be presented as a hyperlink or hypergraphic on this input form. The provider responds by submitting this input form (step 5012). This executes the scheduling method 141 which tests the input form data to see if the provider wishes to take the call immediately (step 5013). If not, the scheduling method 141 tests the input form data to determine if the provider has choosen an alternate time, for example scheduling the call to take place 10 minutes from now (step 5014). If not, the scheduling method 141 determines if the provider wishes to schedule the phone call using automatic scheduling (step 5015). If so, the scheduling method 141 proceeds with autoscheduling as described below starting at step 5070 of FIG. 46. If the provider has chosen to take the call, either immediately or at a later time, the scheduling method 141 next creates a schedule object 110 representing the scheduled phone call (step 5016). The scheduling method 141 saves this schedule object 110 in the provider database 11 (step 5017), then creates a message object 110 containing this schedule object 110 (step 5018). If the phone call request was denied for any reason, this message object 110 will contain a negative acknowledgment message. Lastly the scheduling method 141 transmits this message object 110 to the provider's communications object 110 at the consumer program 22 (step 5019). When received by the consumer program 22, the message object 110 executes a corresponding scheduling method 141 (step 5020). This scheduling method 141 first tests the message object 110 to see if it contains a schedule object 110, meaning the phone call request was accepted (step 5021). If so, the scheduling method 141 saves this schedule object 110 in the consumer database 21 (step 5022). The scheduling method 141 then tests the schedule object 110 to determine if the phone call should being immediately (step 5023). If so, the scheduling method 141 tests the consumer's scheduling preferences to see if the phone call should be autodialed by the consumer program 22 (step 5024). If so, the scheduling method 141 executes a transmission method 141 to autodial the provider's telephone number included in the schedule object 110 (step 5025). The phone call can take place via a public telephone network, via a private phone network, via the Internet, or via any other phone transmission network, as discussed in the transmission control section. Finally, the scheduling method 141 executes the consumer's notification method 141 for initiating phone calls (step 5026). If the phone call was not accepted in step 5021, or the phone call is not to begin immediately in step 5023, or the phone call is not to be autodialed in step 5024, then a message to this effect would be given to the consumer in step 5026.

Detailed Description Text (457):
When either the provider or the consumer are not ready to proceed with a phone call immediately, schedule objects 110 can automate the process of scheduling a future phone call. The steps in this process are illustrated in FIG. 46. This process can be used as an adjunct to the phone call request process described in FIG. 45, or it can be used without a prior call request. The process begins with the consumer executing a hyperlink or hypergraphic representing a voice call scheduling method 141 on a page within the provider's communications object 110 (step 5051). The scheduling method 141 first queries the consumer database 21 to determine the consumer's own schedule (step 5052). Then the scheduling method 141 tests consumers preferences for voice call scheduling to determine if automatic scheduling should be used (step 5053). If not, the scheduling method 141 generates an input form (step 5054). This input form would present the consumer's current schedule, represented by schedule objects 110, and allow the consumer to choose a time to schedule the phone call. The consumer responds by submitting the input form (step 5055). Alternatively, if the consumer choose autoscheduling in step 5053, the scheduling method 141 uses the consumer's scheduling rules 141 and scheduling preferences 147 to determine an optimal schedule time or times for the phone call (step 5056). In either case, once one or more proposed schedule times have been determined, the scheduling method 141 saves the corresponding schedule objects 110 in the consumer database 21 (step 5057). These schedule objects 110 may have a "proposed" attribute to indicate their unconfirmed status. The scheduling method 141 now creates a message object 110 containing the proposed schedule object or objects 110 (step 5058). The scheduling method 141 transmits this to the provider's communications object 110 at the provider program 12 (step 5059). When received by the provider program 12, the message object 110 executes a corresponding scheduling method 141 (step 5060). This scheduling method 141 queries the provider's schedule objects 110 to determine the provider's schedule (step 5061). The scheduling method 141 first tests the result set to see if there are any matching schedule time slots between the consumer's proposed schedule objects 110 and the provider's schedule objects 110 (step 5062). If so, the scheduling method 141 next tests the provider's scheduling preferences to determine if the provider wishes to manually confirm the optimal time match (step 5063). If not, the scheduling method 141 tests to determine if the provider desires notification of the new scheduled item (step 5064). If so, the scheduling method 141 executes the notification method 141 the provider has assigned to new schedule items (step 5065). This notification action may vary with the UID of the consumer, the type of event, the duration of the event, and/or other scheduling or notification rules 140 or preferences 147 established by the provider. If there were no schedule matches in step 5062, the scheduling method 141 checks the provider's scheduling preferences to determine if the provider wishes to proceed with automatic scheduling (step 5066). If not, the scheduling method 141 executes the provider's designated notification method 141 for unfulfilled call scheduling requests (step 5067). For purposes of this example, we will assume this notification method 141 generates an input form (step 5068). This input form can present the proposed schedule times, the provider's conflicts, proposed alternatives, and so on. The provider responds by selecting one or more proposed schedule times and submitting this input form (step 5069). If the automatic scheduling alternative was chosen in step 5066, the scheduling method 141 uses the provider's scheduling rules 141 and/or scheduling preferences 147 to determine an optimal time or times for the phone call (step 5070). Once the schedule time or times are determined in step 5062, 5069, or 5070, the scheduling method 141 saves the corresponding schedule objects 110 in the provider database 11 (step 5071). These object's status attribute can indicate whether the schedule time is proposed or confirmed. The scheduling method 141 then creates a message object 110 containing the schedule object or objects 110 (step 5072). The scheduling method 141 transmits this message object 110 to the provider's communications object 110 at the consumer program 22 (step 5073). When received by the consumer program 22, the message object 110 executes a corresponding scheduling method 141 (step 5074). This scheduling method 141 first queries the consumer's schedule objects 110 to determine the consumer's current status (step 5075). This step is

necessary as the consumer's status may have changed since the last transmission. The scheduling method 141 then tests the result set to see if there are any matching schedule time slots between the provider's schedule objects 110 and the consumer's schedule objects 110 (step 5076). If so, the scheduling method 141 next tests the consumer's scheduling preferences to determine if the provider wishes to manually confirm the optimal time match (step 5077). If not, the scheduling method 141 tests to determine if the consumer desires notification of the new scheduled item (step 5078). If so, the scheduling method 141 executes the notification method 141 the consumer has assigned to new schedule items (step 5079). If there were no schedule matches in step 5076, the scheduling method 141 checks the consumer's scheduling preferences to determine if the consumer wishes to continue with automatic scheduling (step 5080). The consumer may wish to limit autoscheduling to a specified number of attempts. If not, the scheduling method 141 executes the consumer's designated notification method 141 for unfulfilled call autoscheduling requests (step 5081). For purposes of this example, we will assume this notification method 141 generates an input form (step 5082). This input form can present the proposed schedule times, the consumer's conflicts, proposed alternatives, and so on. The consumer responds by selecting one or more proposed schedule times and submitting this input form (step 5083). If the autoscheduling alternative was chosen in step 5080, the scheduling method 141 uses the consumer's scheduling rules 141 and scheduling preferences 147 to determine an optimal schedule time or times for the phone call (step 5084). Once the schedule time or times are determined in step 5076, 5083, or 5084, the scheduling method 141 saves the corresponding schedule objects 110 in the consumer database 11 (step 5085). The scheduling method 141 next tests to determine if the scheduling process is complete (step 5086). If not, the process is repeated starting with step 5058 (step 5088).

Detailed Description Text (458):
Once a phone call has been scheduled, both the provider and consumer have copies of the same schedule object 110 in their respective databases 11, 21. Should either need to reschedule the phone call appointment, they need only edit the schedule object 110. Any changes will be automatically transmitted to the other party. Each can assign his/her own notification methods 141 to the schedule object 110 in order to receive notification of changes in exactly the manner each prefers. The schedule object 110 also provides a channel for communications related to the scheduled phone call. For example, one party could add a message element (211, FIG. 4) containing an agenda for the phone call, and the other would receive it automatically. At the time of the phone call, any elements 143 or other related communications objects or object components could be recalled automatically by the respective schedule objects 110. Additionally, the respective schedule objects 110 can also using event tracking control and reporting control to log the phone call and prepare call reports.

Detailed Description Text (462):
Communications objects 110 encapsulate communications data, metadata, and instructions in order to create an automated communications relationship between the provider and consumer. Additionally, these sets of data, metadata, and instructions are automatically maintained by the communications system of the present invention. Therefore, the databases 11, 21, and 1301 of communications objects represent an attractive central repository for any communications data that must be maintained by the provider or consumer on their computers 1, 2, or 1302, or on a local area network to which those computers are connected. To access the data, metadata, and methods of the communications objects 110 stored in these databases, an applications programming interface (API) can be used. An API defines the methods and method parameters that are used to request services from another application within a desktop, server, or network operating environment. In a preferred embodiment, an API for a communications object system would specify the object services available from a communications object system program 12, 22, or 1302 in a format compatible with other industry standards for distributed object system specifications. Examples of such standards include the DCE and CORBA specifications

from the Open Software Foundation and the OLE and DCOM specifications from Microsoft Corporation.

Detailed Description Text (463):
When the provider and consumer programs 12, 22 include an API, other computer applications can be relieved of the burden of storing, indexing, and maintaining communications data. For instance, the consumer does not need separate address books for a personal information manager, a network directory, and an e-mail program. Other applications can also use the API to automate communications operations using the methods 141 and rules 140 stored in the databases 11, 21, and 1301. A specific example of API usage is the word processing file transfer process explained in the encoding control section and illustrated in FIG. 21.

Detailed Description Text (465):
As explained above, the programs 12, 22, 1302 in a communications object system may also employ a user interface native to the operating system on which the program is run. Many computer operating systems, such as Microsoft Windows from Microsoft Corporation, Apple Macintosh from Apple Computer Corporation, and UNIX Motif from the Open Software Foundation, provide for graphical user interfaces. The use of graphical user interfaces for computer programs is discussed generally in Alan Cooper, About Face (1994), which is incorporated herein by reference. A graphical user interface is advantageous to a communications object system because it allows a user of the programs 12, 22, 1302 to easily and intuitively manipulate visual screen objects in order to create, edit, or delete the underlying communications objects and object associations in the databases 11, 21, 1301. When used in conjunction with a communications object system, this represents a uniquely powerful new interface technique for initiating and controlling communications.

Detailed Description Text (467):
The author palette window 5121 allows the user to visually select different graphical icons for the purpose of creating or editing different types of communications objects or communications object components. Examples shown include a personal communications icon 5122, representing an individual user communications object 110; a group communications icon 5123, representing a user group communications object 110; a telephone number icon 5124, representing a telephone number element 143; a news topic icon 5125, representing a notification element (201, FIG. 4); an open discussion icon 5126, representing a composite topic object (1451, FIG.29B) with a distribution list component to which additional recipients 120 can be added; a closed discussion icon 5127, representing a composite topic object (1451, FIG.29B) to which additional recipients 120 cannot be added; a response icon 5128, representing a component response thread object (1461, FIG. 29B) for adding a response in a discussion group; and a meeting icon 5129, representing a schedule object 110 for setting up a meeting.

Detailed Description Text (468):
The user palette window 5131 allows the user to choose different screen icons representing other communications object system users in the databases 11, 21, or 1301. A user palette allows the user to quickly and easily create recipient associations (121, FIG. 3) or relationship associations (111, FIG. 6B) or to take actions on other user objects. Examples shown include three individual user icons Alice 5132, Bob 5133, and Trent 5134; and three user group icons Marketing 5135, Sales 5136, and Development 5137.

Detailed Description Text (470):
The workspace window 5141 represents a screen area into which author objects and consumer objects can be copied using a mouse or other pointing device in order to manage a specific set of communications relationships. This technique, often called "drag-and-drop", is employed in most operating systems using graphical user interfaces, including those mentioned above. The technique is specifically employed in software programs that employ object-based visual editing for drawing or forms

creation, such as Visio from Visio Corporation and Visual Basic from Microsoft Corporation. In a communications object system user interface, drag-and-drop operations can be used for creating, editing, associating, dessociating, or deleting communications objects and communications object components. For example, a user could create a new open discussion topic 110 by dragging the open discussion icon 5126 into a workspace window 5141. The dragging action would result in a dialog box prompting the user for the new properties of the discussion topic object (1451, FIG.29B). The resulting icon 1542 would then be ready for use. The user could then add other communications object system users to this discussion, such as Mary 5146 and Trent 5147, by dragging by dragging their icons from the user palette 5131 and dropping them on top of the discussion group icon 5126. The other examples shown in the workspace 5141 include a closed discussion 5143, two scheduled meetings 5144 and 5145, and a user group 5148 that has been added to one of the discussions 1542, 1543.

Detailed Description Text (471):
The user can maintain multiple workspace windows 5141 pertaining to each of the user's areas of communications interests. For example, a user could organize workspaces by projects, by departments, by priority, and so on. Workspace windows 5141 can be represented in the databases 11, 21, 1301 by folders (115, FIG. 3) or another separate workspace object class. Workspace windows 5141 can also be represented by communications objects 110 in order that multiple users can easily share the same workspace. In this case changes to one user's workspace 5141 would be reflected in the workspaces of all other users who were consumers of this workspace 5141. This same technique can be applied to object palettes 5121 and user palettes 5131.

Other Reference Publication (11):
D. Woelk, W. Kim & W. Luther "An Object-Oriented Approach to Multimedia Database"ACM 1986.

Other Reference Publication (55):
K. Smith and S. Zdonik "Intermedia: A Case Study of the Differences Between Relational and Object-Oriented Database Systems" OOPSLA '87 Proceedings.

Other Reference Publication (60):
Budi Yuwono and Dik Lun Lee, "Wise: A World Wide Web Resource Database System", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. Aug. 1996.